UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN:
ESPECIALIDAD TELEMÁTICA

PROYECTO FIN DE CARRERA

Aplicación Web para coordinación de reuniones

Autora: Marta Santiago López
Tutor: Vicente Luque Centeno
Junio de 2009

AGRADECIMIENTOS

A mis padres y a mi hermano, que durante años han compartido conmigo tanto mis alegrías como mis desilusiones, y siempre me han apoyado en todo.

A Antonio, que si no hubiera insistido tanto en lo importante que era dar este último paso, lo hubiera dejado hace mucho y no estaríamos hoy aquí.

A Eduardo, que aunque hace muy poco que nos conocemos, se ha volcado conmigo y me ha ayudado en todo y en más de lo que estaba al alcance de su mano, incluido este proyecto.

A Vicente, por su ayuda y su paciencia. Sé que se ha alargado más de lo que los dos hubiéramos querido.

A Manolo y a Sara, que han sido mi brújula.

Y finalmente a todos mis amigos y el resto de mi familia, a la gente que ya no está a mi lado pero que me apoyó en su momento, al resto de los profesores de la universidad que tantas cosas me han enseñado….sin ellos tampoco hubiera podido llegar hasta aquí.

RESUMEN

El objetivo de este proyecto de fin de carrera es diseñar y desarrollar una aplicación de acceso vía Web (mediante un navegador) que permita gestionar las reuniones de un grupo de personas.

Basándose en dos aplicaciones que ya existen, se ha intentado crear un sistema accesible vía HTTP que permita a las personas que necesiten convocar reuniones poder llevarlas a cabo mediante la adquisición de datos de todos los implicados en las mismas. De este modo el grupo de participantes, tanto el convocante como los convocados, podrán acordar entre ellas las horas y fechas pertinentes para poderse reunir fijando igualmente la duración del evento.

Todo el mecanismo de puesta en común de horarios, duraciones y localizaciones del evento se basa en el uso de un navegador y del correo electrónico lo que permite a los diferentes asistentes acceder a la aplicación y estar correctamente informados sobre el resultado final de la convocatoria.

A la aplicación Rendez-Vous se puede acceder de manera temporal en la URL

http://marta.agenciaefe.org:8088/pfc2

mediante el usuario **invitado** y contraseña **telematica**. Este usuario posee permisos de administrador para que de este modo el tribunal pueda ver todas las funcionalidades de las que dispone Rendez-Vous, por lo que se ruega que se trate con la cautela correspondiente a este tipo de usuario.

INDICE GENERAL

INDICE DE FIGURAS

Capítulo 1

Introducción

1.1 MOTIVACIONES

En la actualidad existen muchos programas y elementos electrónicos que permiten gestionar la agenda de una persona, ya sea por motivos profesionales, sociales o de otra índole que dicha persona pueda tener a lo largo del día en cualquiera de las muchas actividades en que se pueda verse envuelto. Sin embargo a menudo es normal que un grupo de personas tenga que coordinarse para poderse citar en una reunión, lo que inevitablemente causará problemas de disponibilidad en cada uno de los asistentes. Cada persona tiene su propia agenda que tiene que poder compatibilizar a la hora de poner llegar a un acuerdo con el resto con objeto de poder acordar el día, hora, duración y lugar de una reunión.

Aunque poco a poco van apareciendo en el mercado aplicaciones que permiten gestionar las disponibilidades de los asistentes a una reunión, suele ser bastante común que estos grupos de personas recurran a una solución socorrida como es el uso de una lista de correos electrónicos para acordar con los distintos participantes los detalles y preferencias con objeto de intentar llegar a un acuerdo.

Con el diseño de esta aplicación, se ha pretendido aunar todas aquellas características de estas aplicaciones que un usuario podría necesitar para planear las reuniones de su agenda y de una forma fácil y sencilla poder ver como encajar su agenda con la de las distintas personas participantes en un evento.

De este modo se consigue sencillez tanto para el organizador como para los invitados a la hora de elegir y gestionar las fechas de las reuniones.

Por último, otro objetivo que se ha establecido con el diseño de esta aplicación es que sea desarrollada, implementada y ejecutada con componentes de código libre, dejando así la aplicación no sólo abierta a posibles ampliaciones y mejoras del proyecto, sino que además se garantice un coste cero en ello.

1.2 OBJETIVOS

Los objetivos a la hora de realizar la aplicación Web, desde ahora ***Rendez-Vous*** fueron:

1. Acceso vía Web con cualquier navegador y con cualquier Sistema Operativo.
2. GUI sencilla.
3. Cumplimiento de las normas establecidas conforme a XHTML 1.1 y nivel de accesibilidad de la herramienta AA.
4. Realizar el desarrollo basándose en componentes de código libre.

Por otra parte, y de una forma más centrada en la funcionalidad de Rendez-Vous, se establecieron los siguientes objetivos:

1. Facilidad a la hora de organizar e invitar gente a una reunión.
2. Facilidad a la hora de expresar las disponibilidades para dicha reunión.
3. Poder proporcionar tantas opciones como se desee a la hora de proponer fechas para una reunión.
4. Posibilidad de contemplar una respuesta por parte de un invitado que no sea ni afirmativa ni negativa.
5. Proporcionar distintas interfaces Web mediante el uso de CSS para demostrar la flexibilidad del producto, algunas de índole corporativa, pero con posibilidad de usar otras más estéticas.
6. Posibilidad de envío de notificaciones, ya sea para avisar de la creación de una nueva reunión o para recordarla, mediante un servidor SMTP y correos electrónicos.
7. Posibilidad de que las claves sean encriptadas para dar mayor seguridad a la hora de almacenar las contraseñas de los usuarios o de las URL de referencia.
8. Permitir a los invitados poner sus preferencias estén o no estén dados de alta en el sistema.
9. Posibilidad de exportar las reuniones pendientes a un fichero de formato ICalendar cuya extensión es .ics, ya sean aquellas reuniones creadas por el propio usuario como a las que ha sido invitado. De este modo aplicaciones como ICal de Mac o Outlook de Microsoft serán capaces de importarlas en sus respectivos calendarios. Así mismo, importar ficheros .ics que generen estos programas para crear las reuniones.

1.3 ESTRUCTURA DE LA MEMORIA

En este proyecto veremos en detalle el diseño partiendo desde cero de una aplicación Web para gestionar reuniones.

Capítulo 1: Introducción

En este capítulo se presenta la motivación por la cual ha surgido este proyecto, cual es su origen así como los objetivos planteados en el desarrollo del mismo. Finalmente se expone un breve apartado describiendo el contenido de esta memoria.

Capítulo 2: Estado del Arte

En *El Estado del Arte*, se procederá a definir las tecnologías utilizadas para el desarrollo e implementación de la aplicación, y una explicación más o menos extensa de cada una de ellas.

Tras ello se pasa a describir específicamente cada una de las capas del diseño de la aplicación:

Capítulo 3. Diseño de la Base de Datos

Se comienza detallando como se ha diseñado la Base de Datos que almacenará todos los datos referentes a las reuniones con sus fechas, usuarios que las han creado y sus invitados. Paso por paso se explicará las decisiones tomadas para el diseño de cada una de las tablas.

Capítulo 4. Base de Datos

Seguidamente se pasa a describir una capa intermedia en la que también se apoya el diseño Web que son las consultas más importantes que se harán a la base de datos diseñada anteriormente.

Capítulo 5:Componentes de la aplicación

Por último se explica el diseño de la aplicación Web basándose en el patrón de arquitectura de software Modelo-Vista-Controlador.

Capítulo 6. Pruebas, Conclusiones y Trabajos Futuros

A continuación se detalla el entorno de pruebas de la aplicación con las conclusiones obtenidas de este proyecto y los trabajos y mejoras que pueden nacer de aquí.

Finalmente se incluyen en la memoria una serie de anexos de apoyo descritos a continuación:

Apéndice A: Manual de instalación

Mediante el apéndice llamado “Manual de Instalación”, se da ciertos pasos para que la aplicación pueda ser implantada en un entorno como podría ser del Departamento de Telemática

Apéndice B: Introducción a la aplicación

También se describe en esta última parte de la memoria y de una forma más detallada todas las utilidades que tiene Rendez-Vous, ayudándose de capturas de pantallas de la propia aplicación para la explicación de cada una de las opciones.

Bibliografía y enlaces de consulta

Para concluir, se incluye un apartado con toda la bibliografía utilizada a lo largo del desarrollo de este proyecto

Capítulo 2

Estado del Arte

En este capítulo se van a explicar las tecnologías que se han usado para implementar la aplicación Rendez-Vous. Como base de la elección, se ha decidido que estas sean de código abierto, ya que no tienen un coste añadido de utilización y cuentan siempre con una gran comunidad de desarrolladores dispuestos a ayudar en futuras mejoras.

En un servidor Apache-Tomcat se ha instalado J2EE que mediante JDBC se conecta a la base de datos MySQL. Dicho motor de base de datos será donde estará organizada toda la información de la aplicación tanto en lo referente a los usuarios como a toda la gestión de las reuniones. Igualmente, la aplicación se apoya en JavaMail como parte imprescindible para mediante correo electrónico poder enviar las notificaciones cuando sea necesario.

Finalmente, es necesario subrayar que la aplicación Web desarrollada cumple los estándares marcado por el XHTML 1.1.

2.1 J2EE

Java Platform Enterprise Edition (J2EE) *[7]* es una plataforma de programación en la que se define el estándar para el desarrollo de aplicaciones empresariales multicapa diseñado por la empresa Sun Microsystems.

J2EE simplifica estas aplicaciones empresariales basándolas en componentes modulares y estandarizados ejecutándose sobre un servidor de aplicaciones, proveyendo un completo conjunto de servicios a estos componentes, y manejando muchas de las funciones de la aplicación de forma automática, sin necesidad de una programación compleja.

Un componente es una unidad de software independiente que forma parte de una aplicación J2EE, con sus clases y ficheros relacionados, comunicándose con otros componentes. En la especificación hay definidos tres tipos:

- Componentes de cliente: aplicaciones instaladas en el cliente y los applets.
- Componentes de Red: Java Servlets y Java Pages (JSPs).
- Componentes de Negocio: Enterprise Java Beans.

Estos componentes se escriben en lenguaje Java, y se compilan de la misma forma que una aplicación cualquiera, pero con la diferencia de que se ensamblan en una aplicación J2EE, se verifica si están bien formados y cumplen la especificación, y se despliegan cuando se ejecuta el servidor.

La última versión de Java EE 5 ha intentado facilitar el desarrollo de sus componentes haciendo uso de la sintaxis del lenguaje, las cuales han sido incluidas en J2SE 5.0. Java 2.1 apoya este objetivo definiendo las anotaciones de inyecciones de dependencias en manejadores de etiquetas JSP y *escuchadores* de contexto.

Otro punto mejorado de Java EE 5 ha sido la especificación de sus tecnologías Web, llamadas JavaServer Pages (JSP), JavaServer Faces (JSF), y JavaServer Pages Standard Tag Library (JSTL).

2.1.1 JSP:

La tecnología de las páginas JSP *[7]* ayuda a crear contenido Web dinámico de una forma rápida y sencilla como respuesta a una petición de un cliente Web. JSP permite un desarrollo rápido de aplicaciones basadas en Web independientemente de servidores y plataformas. JSP se encuentra en la plataforma Java EE 5.

Las páginas JSP se dividen en dos subcategorías:

- Tag Libraries.
- JavaServer Pages Standard Tag Library.

Que son usadas para proveer una plataforma independiente que extiende las capacidades de un servidor Web.

Las principales características que se añadieron a esta definición fueron:

- Soporte para la comprobación de expresiones regulares, que son evaluadas por un gestor de etiquetas cuando es necesario (compilación del archivo), mientras que sus expresiones regulares son evaluadas dinámicamente cada vez que una página es ejecutada.

- Soporte para expresiones Ivalue, que aparecen en la parte izquierda de la operación de asignación. Cuando se usa un Ivalue, una Expresión de Lenguaje EL representa la referencia a una estructura de datos, como por ejemplo podría ser una propiedad de un JavaBean, que ha sido asignado a alguna salida del usuario.

2.1.2 Servlets:

Un Servlet *[6]* es un componente que forma parte de una aplicación Web, y se guarda y ejecuta en un servidor J2EE. Su función principal es interaccionar con el cliente recibiendo peticiones y generando respuestas a partir de ellas.

La tecnología Servlet de Java provee desarrollos Web con un mecanismo simple y consistente para extender la funcionalidad de un servidor Web y un acceso a un sistema de datos. Un Servlet puede ser casi considerado como una applet que corre sobre un servidor. Por ello, los Servlets de Java hacen posibles muchas aplicaciones Web actuales.

La tecnología Servlet proporciona las mismas ventajas del lenguaje Java en cuanto a portabilidad ("*write once, run anywhere*") y seguridad, ya que un Servlet es una clase de Java igual que cualquier otra, y por tanto tiene en ese sentido todas las características del lenguaje.

Además, los Servlets se benefician de la gran capacidad de Java para ejecutar métodos en ordenadores remotos, para conectar con bases de datos, para la seguridad en la información, etc. Se podría decir que las clases estándar de Java ofrecen ya resueltos muchos problemas que con otros lenguajes tiene que resolver el programador.

CARACTERÍSTICAS DE LOS SERVLETS

Las características de los Servlets son:

1. Son independientes del servidor utilizado y de su sistema operativo, lo que quiere decir que al estar escritos en Java, el servidor puede estar escrito en cualquier lenguaje de programación, obteniéndose exactamente el mismo resultado que si lo estuviera en Java.

2. Los Servlets pueden llamar a otros Servlets, e incluso a métodos concretos de otros Servlets. De esta forma se puede distribuir de forma más eficiente el trabajo a realizar. Por ejemplo, se podría tener un Servlet encargado de la interacción con los clientes y que llamara a otro Servlet para que a su vez se encargara de la comunicación con una base de datos. De igual forma, los Servlets permiten redireccionar peticiones de servicios a otros Servlets (en la misma máquina o en una máquina remota).

3. Los Servlets pueden obtener fácilmente información acerca del cliente (la permitida por el protocolo HTTP), tal como su dirección IP, el puerto que se utiliza en la llamada, el método utilizado (GET, POST... etc.).

4. Permiten además la utilización de cookies y sesiones, de forma que se puede guardar información específica acerca de un usuario determinado, personalizando de esta forma la interacción cliente-servidor. Una clara aplicación es mantener la sesión con un cliente.

5. Los Servlets pueden actuar como enlace entre el cliente y una o varias bases de datos en arquitecturas cliente-servidor de 3 capas (si la base de datos está en un servidor distinto).

6. Asimismo, pueden realizar tareas de proxy para un applet. Debido a las restricciones de seguridad, un applet no puede acceder directamente por ejemplo a un servidor de datos localizado en cualquier máquina remota, pero el Servlet sí puede hacerlo de su parte.

7. Al igual que los programas CGI, los Servlets permiten la generación dinámica de código HTML dentro de una propia página HTML. Así, pueden emplearse Servlets para la creación de contadores, banners, etc.

2.1.3 Java Beans

La especificación 1.1 de JavaBeans Enterprise (EJB) *[7]* define una arquitectura para el desarrollo y despliegue de aplicaciones basadas en objetos distribuidos transaccionales, software de componentes del lado del servidor. Estos componentes del lado del servidor, llamados beans, son objetos distribuidos que están localizados en contenedores de JavaBean Enterprise y proporcionan servicios remotos para clientes distribuidos a lo largo de la red.

La especificación de JavaBeans Enterprise define una arquitectura para un sistema transaccional de objetos distribuidos basado en componentes. La especificación manda un modelo de programación; es decir, convenciones o protocolos y un conjunto de clases e interfaces que crean el API EJB. Este modelo de programación proporciona a los desarrolladores de beans y a los vendedores de servidores EJB un conjunto de contratos que definen una plataforma de desarrollo común. El objetivo de estos contratos es asegurar la portabilidad a través de los vendedores y el soporte de un rico conjunto de funcionalidades.

Los beans son componentes de software que se ejecutan en un entorno especial llamado un contenedor EJB. El contenedor contiene y maneja un bean de igual forma que el Servidor Web Java contiene un servlet o un servidor Web contiene un applet Java. Un bean no puede funcionar fuera de un contenedor EJB. El contenedor EJB controla cada aspecto del bean en tiempo de ejecución incluyendo accesos remotos al bean, seguridad, persistencia, transacciones, concurrencia, y accesos a un almacén de recursos.

El contenedor aísla al bean de accesos directos por parte de aplicaciones cliente. Cuando una aplicación cliente invoca un método remoto de un bean Enterprise, el contenedor primero intercepta la llamada para asegurar que la persistencia, las transacciones, y la seguridad son aplicadas apropiadamente a cada operación que el cliente realiza en el bean. El contenedor maneja estos aspectos de forma automática, por eso el desarrollador no tiene que escribir este tipo de lógica dentro del propio código del bean. El desarrollador de beans puede enfocarse en encapsular las reglas del negocio, mientras el contenedor se ocupa de todo lo demás.

Los contenedores manejan muchos beans simultáneamente de igual forma que un Java WebServer maneja muchos Servlets. Para reducir el consumo de memoria y de proceso, los contenedores almacenan los recursos y manejan los ciclos de vida de todos los beans de forma muy cuidadosa. Cuando un bean no está siendo utilizado, un contenedor lo situará en un almacén para ser reutilizado por otros clientes, o posiblemente lo sacará de la memoria y sólo lo traerá de vuelta cuando sea necesario. Como las aplicaciones cliente no tienen acceso directo a los beans --el contenedor trata con el cliente y el bean-- la aplicación cliente se despreocupa completamente de las actividades de control de recursos del contenedor. Por ejemplo, un bean que no está en uso, podría ser eliminado de la memoria del servidor, mientras que su referencia remota en el cliente permanece intacta. Cuando un cliente invoca a un método de la referencia remota, el contenedor simplemente re-activa el bean para servir la petición. La aplicación cliente se despreocupa de todo el proceso.

2.2 JDBC

JDBC *[9]* es una especificación de un conjunto de clases y métodos de operación que permiten a cualquier programa Java acceder a sistemas de bases de datos de forma homogénea. La aplicación de Java debe tener acceso a un controlador (driver) JDBC adecuado. Este controlador es el que implementa la funcionalidad de todas las clases de acceso a datos y proporciona la comunicación entre el API JDBC y la base de datos real. De manera muy simple, al usar JDBC se pueden hacer tres cosas:

- Establecer una conexión a una fuente de datos (ej. una base de datos).
- Mandar consultas y sentencias a la fuente de datos.
- Procesar los resultados.

Toda la conectividad de bases de datos de Java se basa en sentencias SQL, por lo que se hace imprescindible un conocimiento adecuado de SQL para realizar cualquier clase de operación de bases de datos.

2.3 JAVAMAIL

JavaMail *[13] [14]* provee una plataforma para proporcionar acceso independiente del protocolo para construir, enviar y recibir mensajes electrónicos y aplicaciones de mensajería. El API de JavaMail es un paquete opcional que se usa con la plataforma Java SE y también esta incluida en la plataforma Java EE.

2.4 MySQL

MySQL *[8] [9]* es un sistema de gestión de bases de datos relacionales, multihilo y multiusuario que utiliza el estándar SQL como lenguaje de acceso. Una base de datos relacional es una colección estructurada de datos dividida en diferentes tablas, para mejorar su velocidad y flexibilidad. Con multihilo se quiere decir que es capaz de atender varias peticiones al mismo tiempo, y el soporte multiusuario nos da la posibilidad de crear varias cuentas en el sistema con diferentes permisos.

El Software de la base de datos de MySQL es un sistema cliente/servidor que consiste en un servidor SQL que soporta diferentes backend, varios programas de cliente y librerias, aplicaciones administrativos y un amplio rango de APIs. Además el servidor MySQL puede emplearse como una librería que se puede integrar en la aplicación para hacerla mas pequeña, mas rápida y un producto autónomo fácil de manejar.

MySQL es la base de datos de código abierto mas popular mundialmente, por lo que se ha convertido en uno de los pilares de las aplicaciones Web construidas con LAMP (Linux, Apache, MySQL, PHP / Perl / Python.). MySQL puede ejecutarse en mas de 20 plataformas incluyendo Linux, Windows, OS/X, HP-UX, AIX, Netware.

El software MySQL tiene una licencia GPL. Al ser de Código Abierto, es posible que cualquier persona lo use y lo modifique. Cualquiera puede descargar el software MySQL de Internet y usarlo sin pagar nada, a diferencia de otras tecnologías alternativas como pueden ser Oracle o Microsoft SQL Server.

MySQL ha sido desarrollada, distribuida, y soportada por MySQL AB, una compañía comercial, fundada por los desarrolladores de MySQL.

2.5 TOMCAT

El servidor Apache Tomcat *[10]* es una implementación de referencia de las tecnologías Java Servlet y JSP. Es un contenedor Web, que administra peticiones de clientes e invoca aplicaciones (formadas por Servlets y JSPs) para satisfacerlas.

Desplegar una aplicación consiste en incluir una serie de ficheros en un contenedor Web (por ejemplo, Tomcat) para que los clientes puedan acceder a su funcionalidad. Normalmente, el desarrollo de un Servlet forma parte de lo que se denomina una aplicación Web, que no es más que una colección de Servlets, páginas HTML, JSP, clases y otros recursos que se pueden empaquetar y ejecutar en distintos contenedores Web, de distintos vendedores, y que ofrecen una determinada funcionalidad a la que los clientes acceden típicamente a través de un navegador.

Las Aplicaciones Web, a partir de la especificación de Servlet 2.2, deben estructurarse según la siguiente jerarquía de subdirectorios:

- Directorio raíz: puedes publicar ficheros estáticos (HTML, imágenes, hojas de estilo, etc.) y JSPs.

    - Directorio WEB-INF: debe contener un fichero web.xml. Este fichero configura la aplicación. Por ejemplo, permite declarar Servlets, asignarles parámetros de inicio, declarar alias y filtros, etc.

    - Directorio classes: puedes colocar en él los ficheros compilados (Servlets, beans, etc.) de las clases utilizadas por la aplicación web.

    - Directorio lib: puedes colocar en él otras bibliotecas de clases adicionales (comprimidas con jar) que utilice tu aplicación.

    - Resto de subdirectorios: para ficheros estáticos y JSP.

Para exportar aplicaciones Web a otros servidores, se puede comprimir esta estructura de directorios, utilizando jar, dando lugar así a un fichero WAR (almacénalo con extensión war, por ejemplo prueba.war). Este fichero es compatible con cualquier plataforma de ejecución de Servlets y JSP.

2.6 XHTML:

XHTML *[15] [16],* acrónimo inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), está pensado para sustituir a HTML como estándar para las páginas web. XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo es lograr que la información, y la forma de presentarla estén claramente separadas.

La necesidad de una versión más estricta de HTML se debió principalmente porque el contenido de la World Wide Web ahora puede visualizarse desde numerosos dispositivos (como móviles) aparte de los ordenadores tradicionales, donde no pueden dedicarse recursos suplementarios para afrontar la complejidad añadida de la sintaxis del HTML.

Ventajas

Las principales ventajas del XHTML sobre otros formatos son:

- Compatibilidad parcial con navegadores antiguos: la información XML puede visualizarse, aunque sin formato. Cabe apuntar que el XHTML 1.0 fue diseñado expresamente para ser mostrado en navegadores que soportan HTML base.
- Un mismo documento puede adoptar diseños distintos en diferentes dispositivos, pudiendo incluso escogerse entre varios diseños para un mismo medio. *[15] [17] [18]*
- Facilidad de edición directa del código y de mantenimiento.
- Formato abierto, compatible con los nuevos estándares que actualmente está desarrollando el W3C como recomendación para futuros agentes de usuario o navegadores.
- Los documentos escritos conforme a XHTML 1.0 pueden potencialmente presentar mejor rendimiento en las actuales herramientas web que aquellos escritos conforme a HTML.

Inconvenientes

- Algunos navegadores antiguos no son totalmente compatibles con los estándares, lo que hace que las páginas no siempre se muestren correctamente. Esto cada vez es menos problemático, al ir cayendo en desuso y ser remplazadas por versiones mas modernas.
- Muchas herramientas de diseño Web aún no producen código XHTML correcto.

2.7 JavaScript

JavaScript *[18] [19]* es un lenguaje que se integra directamente en páginas HTML. Tiene como características principales las siguientes:

- Es interpretado (que no compilado) por el cliente, es decir, directamente del programa fuente se pasa a la ejecución de dicho programa, con lo que al contrario que los lenguajes compilados no se genera ni código objeto ni ejecutable para cada máquina en el que se quiera ejecutar dicho programa.

- Está basado en objetos, pero a diferencia de Java, no emplea clases ni herencia, ni otras técnicas típicas de la OOP.

- Su código se integra en las páginas HTML, incluido en las propias páginas.

- No es necesario declarar los tipos de variables que van a utilizarse, ya que JavaScript realiza una conversión automática de tipos.

- Las referencias a objetos se comprueban en tiempo de ejecución. Esto es consecuencia de que JavaScript no es un lenguaje compilado.

- No puede escribir automáticamente al disco duro. Por eso JavaScript se denomina un lenguaje seguro para el entorno.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas Web. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del DOM (Document Object Model).

2.8 Cifrado de Password mediante el uso de HASH

La mayoría de las páginas Web actuales tienen algún tipo de módulo donde a un usuario se le pide introducir su usuario y contraseña para completar un proceso de autenticación, siendo estos datos almacenados en una base de datos. Para ello, es necesario que el diseñador Web emplee un algoritmo de cifrado lo mas seguro posible de modo que proteja estos datos.

One-way hash encryption:

El escenario que se tiene es un candidato perfecto para el cifrado basado en funciones hash, también conocido como message digest, One-way hash encryption *[11] [12]* , huella digital, o hash criptográfico. Se le denomina de un solo sentido porque a pesar de que puedes calcular un *message digest*, posee cierta información, que hace que sea un proceso irreversible. Además, también es un mecanismo que garantiza que dos valores diferentes no pueden producir el mismo resultado encriptado.
Por otra parte, la representación encriptada obtenida tiene siempre el mismo tamaño.

Los algoritmos hash mas utilizados actualmente son:

Algorithm	**Strength**
MD5	128 bit
SHA-1	160 bit

MD5 (acrónimo de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es un Algoritmo de Reducción Criptográfico de 128 bits ampliamente usado.

SHA-1 (Secure Hash Algorithm 1) es más lento que MD5, pero el resultado de esta función es mas larga (160 bits), lo que le hace más resistente a los ataques por fuerza bruta. Además, es recomendable que se use preferentemente SHA en vez de MD5 por todas las necesidades de seguridad.

Hay que comentar que SHA-1 tiene actualmente evoluciones aun mas fuertes como son SHA-256, SHA-384, y SHA-512, produciendo 256, 384 y 512-bits de codificación respectivamente.

2.9 ICalendar

iCalendar *[20]* define un estándar recogido dentro del RFC 2445 donde se detalla el intercambio de información de calendarios. A este estándar también se le suele denominar "iCal", ya que el programa de Apple Computer del mismo nombre, fue la primera aplicación que implementó dicho estándar.

Gracias a iCalendar los usuarios pueden invitar a reuniones o asignar tareas a otros usuarios a través del correo electrónico. Si el destinatario del mensaje en formato iCalendar tiene un programa capaz de trabajar con este estándar, puede responder fácilmente aceptando la invitación, o proponiendo otra fecha y hora para la misma.

El estándar ha sido ya implementado en una amplia variedad de programas que incluyen como hemos dicho a iCal de Apple, Mozilla Calendar, Google Calendar, Chandler, Lotus Notes, ScheduleWorld, Korganizer, Mulberry, Evolution de Novell, Kronolith, Simple Groupware, Windows Calendar y Microsoft Outlook.

Aunque normalmente la información en formato iCalendar se transmite por correo electrónico, el estándar fue diseñado para ser independiente del protocolo de transmisión de los datos. Por ejemplo, un servidor Web basado exclusivamente en el protocolo HTTP, puede ser usado para distribuir la información de un evento, o indicar las horas y las fechas en que el usuario estará libre u ocupado.

Capítulo 3

Diseño de la Base de Datos

3.1 Diseño del Modelo Entidad-Relación

En principio y para comenzar el desarrollo de la aplicación, se estudiaron de forma detallada las necesidades reales que el usuario final podría tener para gestionar una reunión con los diferentes asistentes.

Para ello, nos basamos en dos aplicaciones Web ya existentes que intentan ayudar a gestionar las reuniones de las personas: MeetingWizard *[1]* y Doodle *[2],* anotando todas las funcionalidades ofrecidas por ellos así como sus carencias para poder crear una herramienta lo más sencilla de usar y al mismo tiempo lo más completa.

Antes de nada, y siguiendo el "Modelo Entidad-Relación" se definieron las 3 entidades principales de la base de datos que son Usuario, Evento, y Fecha, así como sus atributos necesarios.

3.1.1 USUARIO

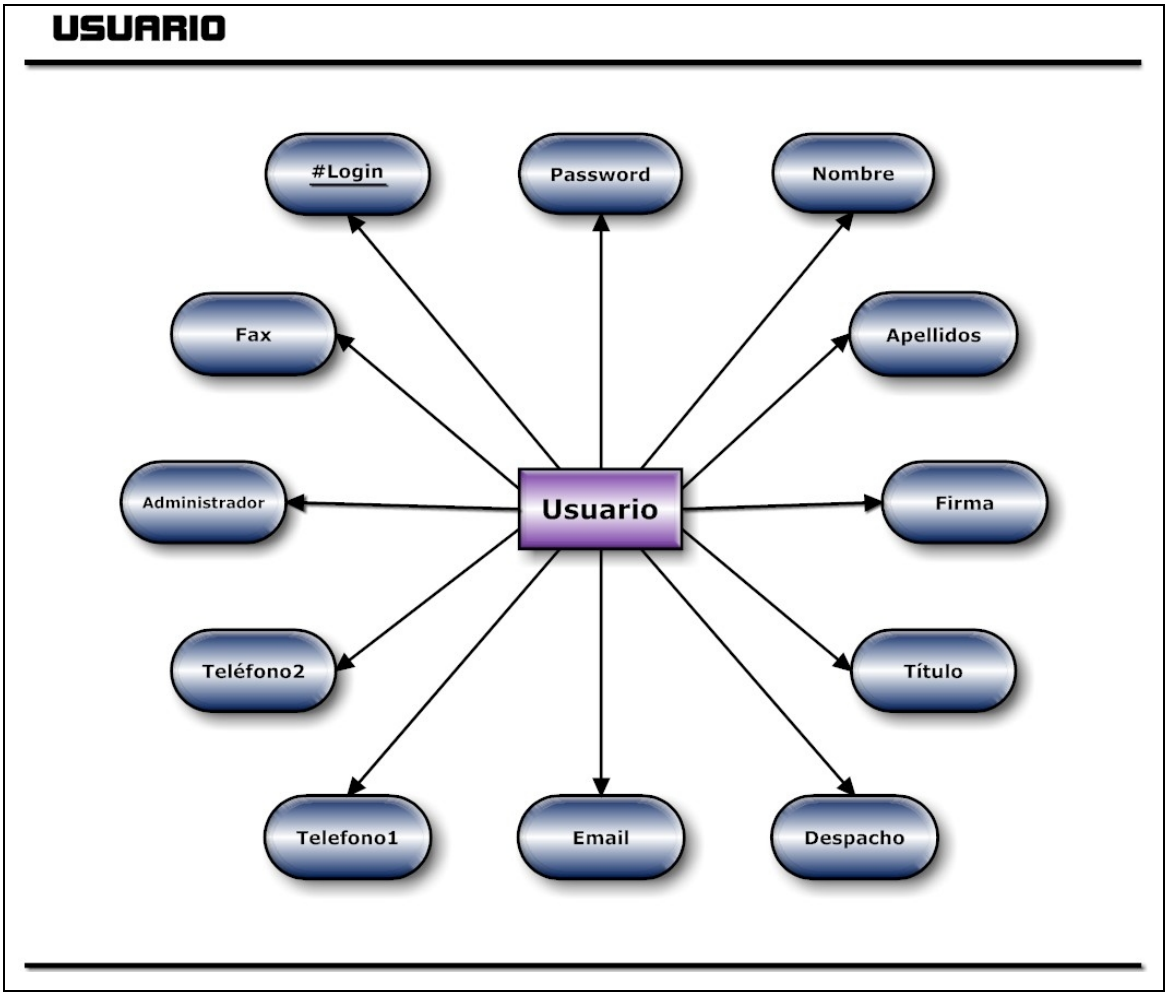

Figura 3.1: Entidad Usuario

La entidad Usuario se usa para almacenar los datos de las personas registradas en la aplicación. Cuenta con doce propiedades que intentan recoger los datos más relevantes de un usuario. Como se puede observar, se almacenan tanto datos identificativos del usuario como datos de contacto.

Además, se incluyen también un atributo llamado **TITULO** que se usa por si el usuario prefiere que se le cite como Sr, Sra., Srta., o incluso Doctor.

Por otra parte, el atributo **FIRMA** se agregará en los correos electrónicos, tal y como indica su nombre, a modo de firma de estos para personalizarlos.

El atributo **ADMINISTRADOR**, de uso de interno de la aplicación, restringe si un usuario puede acceder a la parte de administración de la aplicación pudiendo crear, borrar, consultar y modificar los datos de otros usuarios.

3.1.2 EVENTO

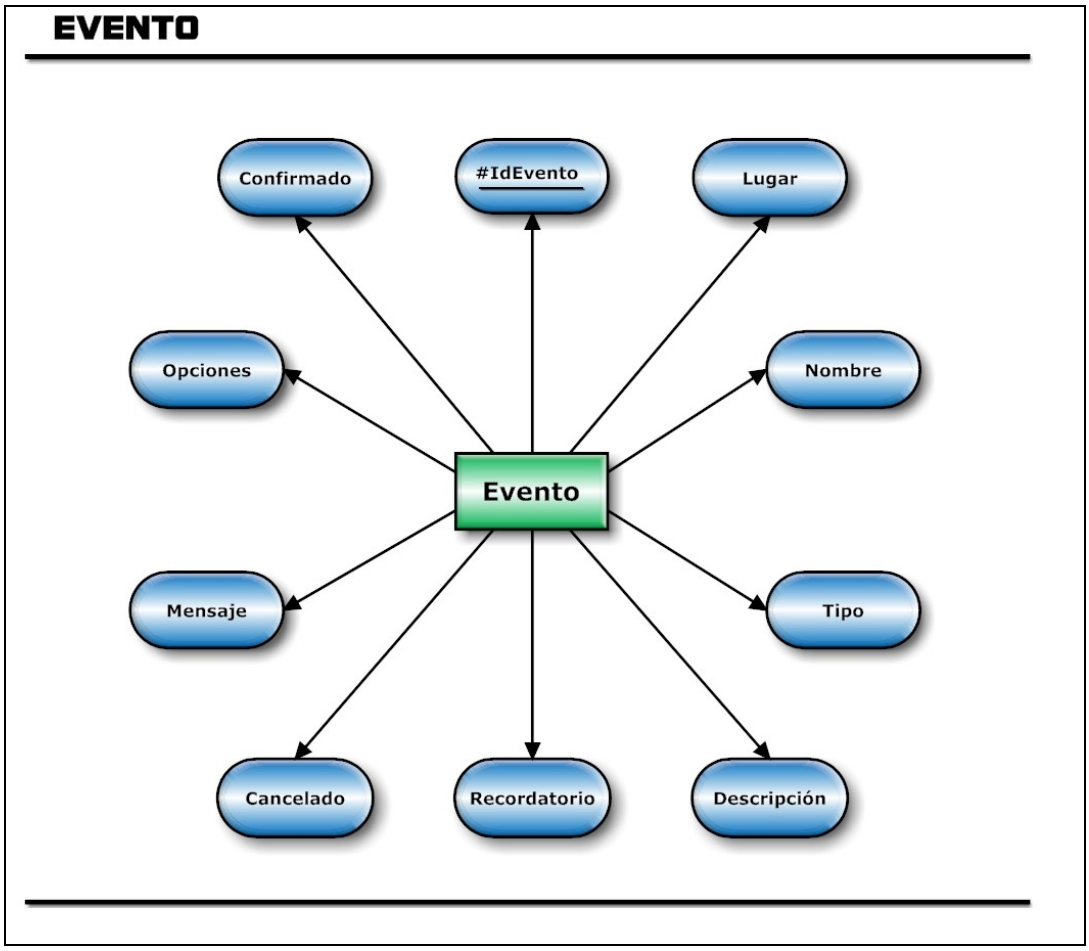

Figura 3.2: Entidad Evento

La entidad Evento se encarga de recoger todos los datos relacionados con cualquier tipo de reunión que se organice con *Rendez-Vous*. Para ello, cuenta con 10 atributos que definen los principales datos que se necesita para describir una reunión.

NOMBRE, **LUGAR** y **DESCRIPCION** intentan definir con una breve explicación el propósito de la reunión.

El **TIPO** de Evento corresponde a un número entero cuyos valores son:

1. - Face-to-face
2. - Teleconference
3. - On line chat
4. - Videoconference
5. - Social
6. - To be determinate

Por otra parte, **OPCION** se usa en caso de que a parte de la descripción, se necesite añadir algún comentario breve mas especifico.

RECORDATORIO, cuyo uso es para poder avisar al organizador de cuando los invitados responden, siendo:

1.- Después de todas las respuestas recibidas.
2.- Cada vez que una respuesta es recibida.

*NOTA: La opción de **recordatorio** y **opción** se encuentra contemplada para futuras mejoras.*

MENSAJE guarda el comentario que hace el organizador de la reunión una vez cree las invitaciones.

Por último, el atributo **CONFIRMADO** indica si esta reunión ya tiene una fecha confirmada para celebrarse.

3.1.3 FECHA

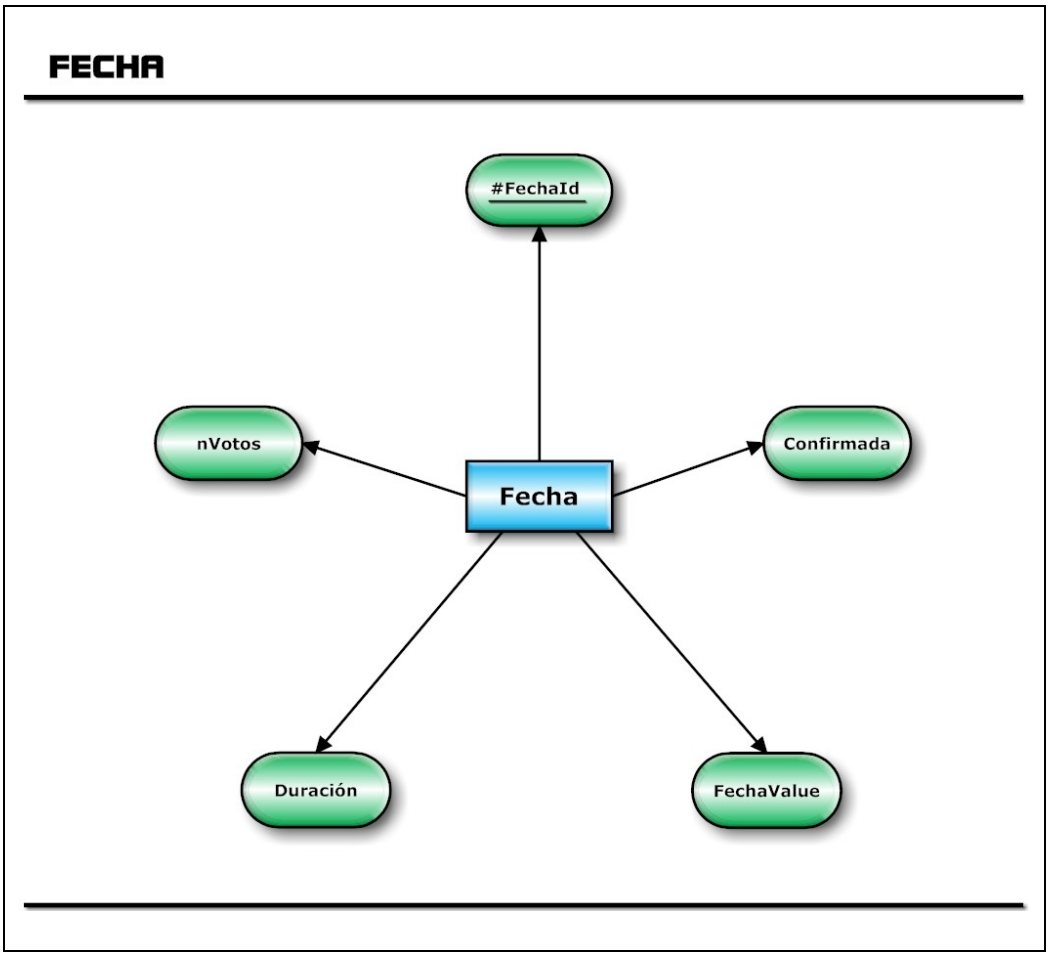

Figura 3.3: Entidad Fecha

La entidad **FECHA**, tal y como su nombre indica, se encarga de recoger todos los datos relacionados a las fechas en las que se desarrollan o se pueden desarrollar una reunión. Para ello cuenta únicamente con 5 atributos.

El atributo **FECHAVALUE** se encarga de recoger la fecha en formato del calendario YYYY-MM-DD.

CONFIRMADA recoge si una fecha es la confirmada para una reunión o no. Es decir, al crear un Evento, éste puede ser confirmado o no como se explicó anteriormente, y las fechas relacionadas a él deben de ser coherentes con ello.

3.1.4 Modelo Entidad-Relación

Finalmente, se definieron todas las relaciones que habría entre ellas, quedando finalmente el diagrama Entidad-Relación de la aplicación *Rendez-Vous* que se ha desarrollado de la siguiente manera:

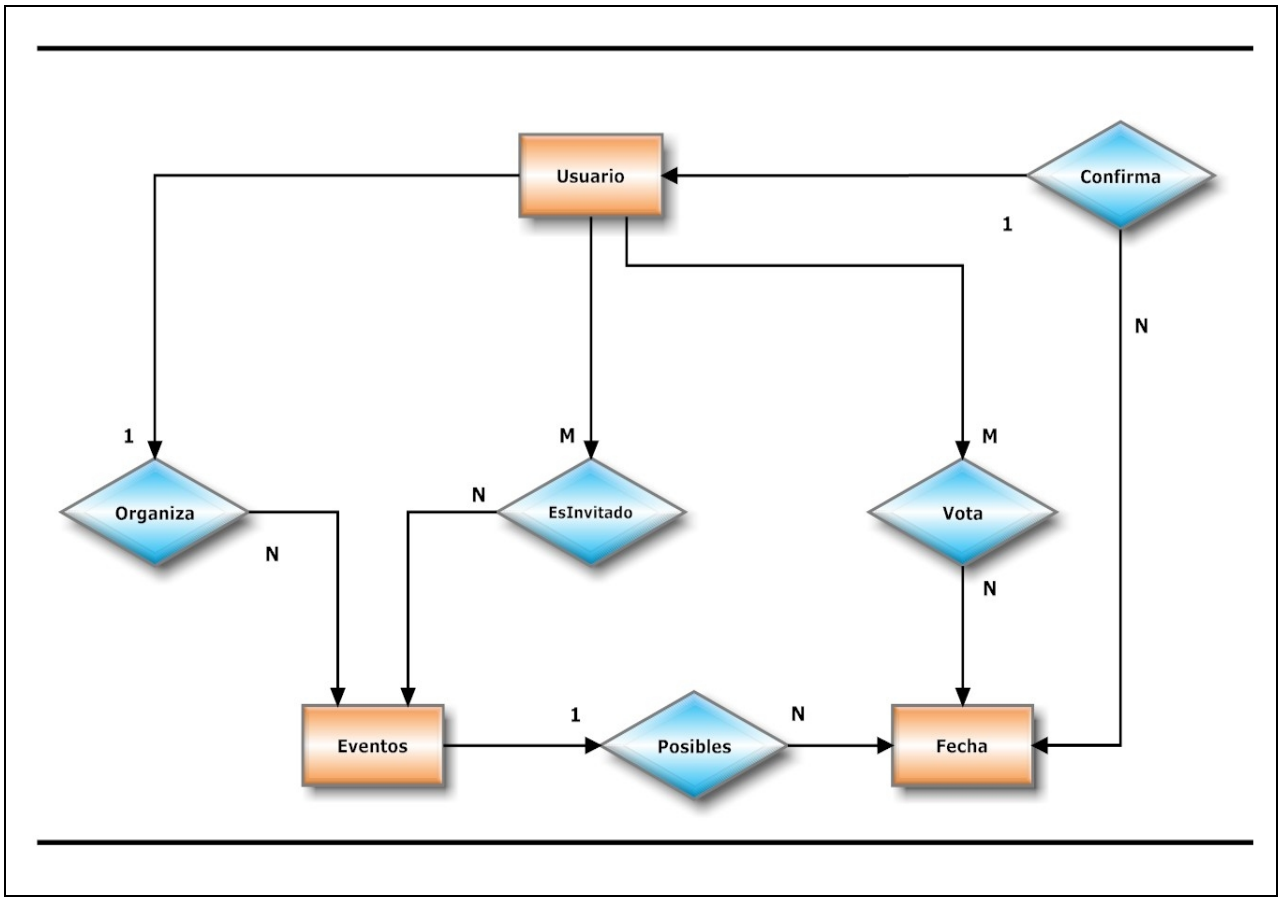

Figura 3.4: Modelo Entidad-Relación

3.2 Diseño de la Base de Datos Relacional

Una vez creado el modelo Entidad-Asociación, se pasó a diseñar la Base de Datos Relacional:

3.2.1.- Relaciones Conjunto Entidad Fuerte -> R

Siguiendo la teoría, *"Por cada conjunto de entidad E fuerte del esquema ER, se crea una relación R, conteniendo los atributos simples de E y eligiendo para cada uno de ellos una clave primaria*".

Usuario (#Login, Nombre, Apellidos, Despacho, Email, Telefono1, Telefono2, Fax, Firma, Administrador)
Evento (#IdEvento, Nombre, Descripción, Lugar, Tipo, Recordatorio, Mensaje, Opciones, Confirmado, Cancelado)
Fecha (#IdFecha, Fechavalue, Confirmado, Duración, Votos)

Tal y como lo hicimos anteriormente.

3.2.2.- Asociación 1:N -> S y T

"Por cada asociación 1:N se identifican las relaciones S y T, siendo S el participante del lado N. Una vez hecho esto, se le añade a S la clave primaria de T, que será clave externa de S y los atributos propios de la relación."

En este primer caso, se identifica como S a Fecha y T a Usuario, y se le añade a **Fecha** la clave primaria del segundo:

Fecha(#IdFecha, FechaValue, Confirmado, Duración, Votos, **#LoginUsuario**)

Figura 3.5: Asociación *Usuario-confirma-Fechas*

Se identifica a **Evento** como T y a Fecha como S, por lo que la relación S queda de la siguiente forma:

Figura 3.6: Asociación *Evento-posibles-Fechas*

Se identifica la entidad de lado N como T, en este caso a **Evento**; a **Fecha**, al ser la entidad de lado 1, como S, por lo que la relación S queda de la siguiente forma:

Fecha (#IdFecha, FechaValue, Confirmado, Duración, Votos, #LoginUsuario, **#IdEvento)**

Por otro lado, en la relación:

Figura 3.7: Asociación *Usuario-organiza-Evento*

Como se explicó anteriormente, **Usuario** es T y **Evento** es S, quedando:

Evento (#IdEvento, Nombre, Descripción, Lugar, Tipo, Recordatorio, Mensaje, Opciones, Confirmado, Cancelado, **#LoginUsuario)**

3.2.3.- Asociación N:M -> R

"*Por cada Asociación N: M se crea una nueva relación R para representar la Asociación N:M, incluyendo en la nueva relación las claves primarias que lo asocian, y a la vez, estas claves constituirán la clave primaria de la Relación*"

En la asociación:

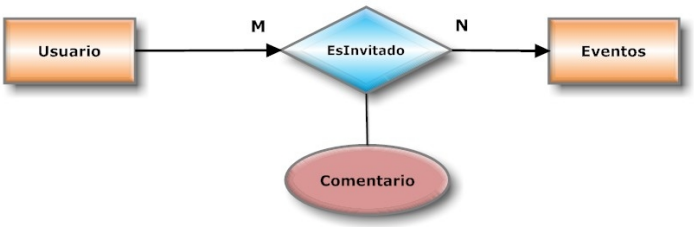

Figura 3.8: Asociación *Usuario-EsInvitado-Evento*

Se crea la nueva relación **EsInvitado**, pero debido a que se puede invitar a gente que no tiene porqué estar dado de alta en la aplicación, se decidió sustituir la clave primaria de Usuario por *EmailInvitado*. Por otra parte, se añade el campo *Comentario* perteneciente a la relación:

EsInvitado(#EmailInvitado, #IdEvento, Comentario)

Ahora en:

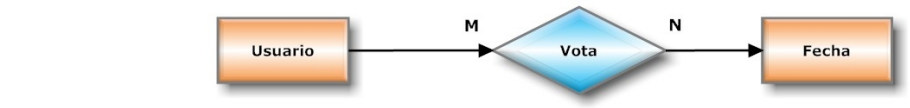

Figura 3.9: Asociación *Usuario-vota-Fecha*

Creamos la Relación **VotaFecha**, y al igual que antes se sustituye la clave primaria de Usuario por **EmailInvitado** por el motivo anteriormente explicado, y se añade también el campo *Respuesta.*

VotaFechas(#EmailInvitado, #IdFecha, Respuesta)

Finalmente, la Base de Datos de Rende-Vous queda de la siguiente forma:

Primero, podemos observar las 5 tablas que previamente habíamos definido:

```
mysql> show tables;
+--------------+
| Tables_in_mw |
+--------------+
| EsInvitado   |
| Evento       |
| Fecha        |
| Usuario      |
| VotaFechas   |
+--------------+
5 rows in set (0.00 sec)
```

Figura 3.10: Tablas de la Base de Datos

A continuación, se define cada tabla en profundidad.

En la tabla **Usuario**, todos los campos son Strings, pero dependiendo de su posible longitud, se ha optado por asignarlos como VARCHAR y CHAR.

Los campos **TELEFONO1**, **TELEFONO2** y **FAX**, si son rellenados, siempre tendrán que tener una longitud de 10 caracteres (aunque su comprobación se realiza en el middleware).

El campo **ADMINISTRADOR** podrá ser *SI* o *NO*, pero por defecto se crea éste a *NO*. Es el único campo que se rellena por defecto.

```
mysql> desc Usuario;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| LOGIN         | char(10)    | NO   | PRI | NULL    |       |
| PASSWORD      | char(100)   | NO   |     | NULL    |       |
| NOMBRE        | varchar(45) | NO   |     | NULL    |       |
| APELLIDOS     | varchar(45) | NO   |     | NULL    |       |
| TITULO        | varchar(45) | YES  |     | NULL    |       |
| DESPACHO      | varchar(45) | NO   |     | NULL    |       |
| EMAIL         | varchar(45) | NO   |     | NULL    |       |
| TELEFONO1     | char(10)    | NO   |     | NULL    |       |
| TELEFONO2     | char(10)    | YES  |     | NULL    |       |
| FAX           | char(10)    | YES  |     | NULL    |       |
| FIRMA         | text        | NO   |     | NULL    |       |
| ADMINISTRADOR | varchar(2)  | NO   |     | NO      |       |
+---------------+-------------+------+-----+---------+-------+
12 rows in set (0.01 sec)
```

Figura 3.11: Descripción de la tabla Usuario

Se puede observar que la clave primaria de los **Eventos** (*IDEVENTO*) se rellenará automáticamente mediante la función de autoincremento de MySQL, aunque para evitar algún posible fallo, se comprueba nuevamente en el middleware.

En la tabla **Evento**, tenemos varios campos que tienen la correspondencia alfanumérica que se definieron anteriormente:

El **TIPO** de Evento:

1.- Face-to-face
2. - Teleconference
3. - On line chat
4.- Videoconference
5. - Social
6. - To be determinate

Y **RECORDATORIO**:

1.- Después de todas las respuestas recibidas.
2.- Cada vez que una respuesta es recibida.

Para evitar posibles malentendidos, los campos **CANCELADO** y **CONFIRMADO** se crean ambos a *NO*, y se deberá cambiar su valor cuando se considere oportuno.

```
mysql> desc Evento;
+--------------+-------------------------------+------+-----+---------+----------------+
| Field        | Type                          | Null | Key | Default | Extra          |
+--------------+-------------------------------+------+-----+---------+----------------+
| NOMBRE       | varchar(45)                   | NO   |     | NULL    |                |
| IDEVENTO     | int(11)                       | NO   | PRI | NULL    | auto_increment |
| DESCRIPCION  | text                          | YES  |     | NULL    |                |
| LUGAR        | varchar(100)                  | NO   |     | ""      |                |
| TIPO         | enum('1','2','3','4','5','6') | NO   |     | 1       |                |
| RECORDATORIO | enum('0','1','2')             | NO   |     | 1       |                |
| MENSAJE      | text                          | NO   |     | NULL    |                |
| OPCIONES     | varchar(45)                   | NO   |     | NULL    |                |
| LOGINUSUARIO | char(10)                      | NO   | MUL | NULL    |                |
| CONFIRMADO   | varchar(2)                    | NO   |     | NO      |                |
| CANCELADO    | varchar(2)                    | NO   |     | NO      |                |
+--------------+-------------------------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

Figura 3.12: Descripción de la tabla Evento

En la tabla **Fecha** es necesario tener claro que la clave primaria no apunta a un posible valor de fechas, sino que es un número identificativo, ya que los valores de cada fecha posiblemente no sean únicos.

Además de esto, los valores de la tabla **Fecha** siguen el patrón de YYYY-MM-DD, por lo que es necesario que a la hora de introducir cualquier valor de fecha esto se cumpla.

Si una **Fecha** es confirmada, su valor almacenado será 1 y si no lo es, será 0.

```
mysql> desc Fecha;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| IDFECHA      | int(11)     | NO   | PRI | NULL    | auto_increment |
| FECHAVALUE   | datetime    | NO   |     | NULL    |                |
| CONFIRMADO   | varchar(1)  | NO   |     | 0       |                |
| LOGINUSUARIO | char(10)    | NO   | MUL | NULL    |                |
| IDEVENTO     | int(11)     | NO   | MUL | NULL    |                |
| DURACION     | varchar(45) | YES  |     | NULL    |                |
| NVOTOS       | int(11)     | NO   |     | 0       |                |
+--------------+-------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

Figura 3.13: Descripción de la tabla Fecha

En **VotaFechas**, nos volvemos a encontrar con otra correspondencia para facilitar su almacenamiento:

En el campo **RESPUESTAS** se tiene que tener en cuenta que:

S: Sí
N: No
NC: No contestado
P: Posiblemente.

En su creación, la respuesta es NC.

```
mysql> desc VotaFechas;
+---------------+-----------------------+------+-----+---------+-------+
| Field         | Type                  | Null | Key | Default | Extra |
+---------------+-----------------------+------+-----+---------+-------+
| EMAILINVITADO | varchar(50)           | NO   | PRI | NULL    |       |
| IDFECHA       | int(11)               | NO   | PRI | NULL    |       |
| RESPUESTA     | enum('S','N','P','NC') | NO   |     | NC      |       |
+---------------+-----------------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Figura 3.14: Descripción de la tabla VotaFechas

Por último, y como dijimos antes, la tabla **EsInvitado** relaciona cada invitado con el evento al que pertenecen, y cualquier posible comentario de éstos.

```
mysql> desc EsInvitado;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| EMAILINVITADO | varchar(45) | NO   | PRI | NULL    |       |
| IDEVENTO      | int(11)     | NO   | PRI | NULL    |       |
| COMENTARIO    | text        | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

Figura 3.15: Descripción de la tabla EsInvitado

Capítulo 4

Base de Datos

Una vez diseñada la arquitectura de la aplicación *Rendez-Vous*, se procedió a diseñar los JavaBeans necesarios para encapsular la información de la Base de Datos y las principales consultas que se iban a realizar sobre ésta. Para realizar este segundo paso, se diseñó un programa por línea de comandos en java que contemplase las funcionalidades más básicas de *Rendez-Vous*.

El resultado de este estudio fue el siguiente:

4.1.- Creación de los JavaBeans que contendrán referencias a la base de datos.

Cada JavaBean, como se muestra a continuación en la Figura 4.1, refleja en sus propiedades los elemento necesarios que se contemplan en cada una de las tablas de la Base de Datos.

A su vez, se definieron seguidamente la relación existente entre cada uno de los Beans, y los atributos que los relacionan entre ellos.

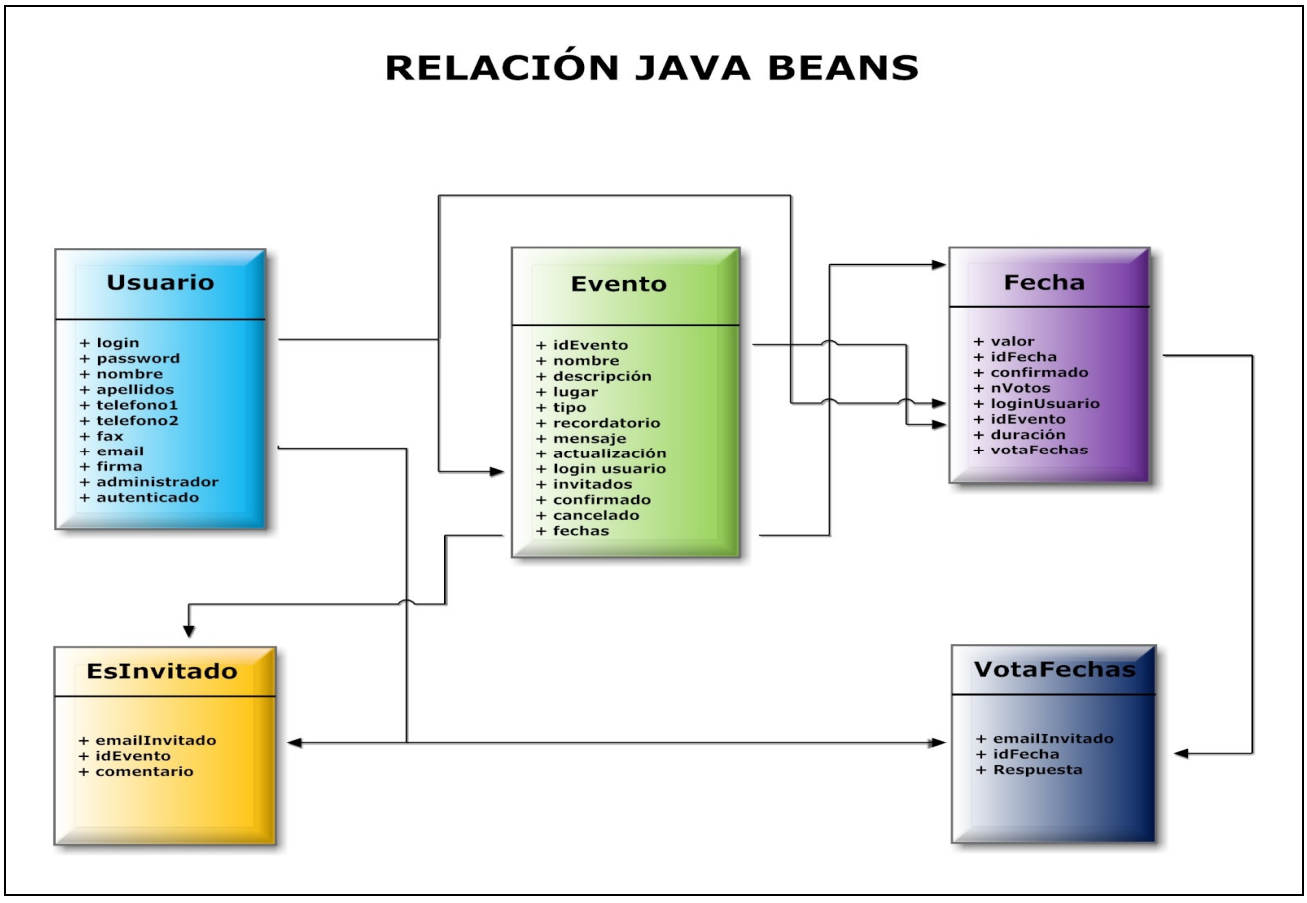

Figura 4.1: Relación de los Beans

4.2 Diseño de las principales consultas sobre la Base de Datos

Una vez hecho esto, se pasó al diseño del programa por línea de comandos donde lo que se pretendía hacer era simular las funcionalidades de Rendez-Vous, definiendo muy precisamente las consultas que serían necesarias hacer a la base de datos, así como la inserción y modificación de los registros según el programa fuera ejecutándose.

4.2.1. - Un usuario se autentica:

Comprobamos si este usuario existe en la base de datos comprobando sobre la tabla usuario el nombre de usuario y la contraseña codificada. Si el resultado es distinto de -1, dicho usuario esta dado de alta y su contraseña es correcta.

```
select * from Usuario where LOGIN='login' and
        PASSWORD='passCodificada';
```

Figura 4.2: Consulta para obtener el usuario mediante nombre de usuario y contraseña

4.2.2.- Creación de un nuevo usuario:

Primero comprobamos que no haya nadie con ese nombre de usuario:

```
select * from Usuario where LOGIN='login';
```

Figura 4.3: Consulta para obtener todos los usuarios con un nombre de usuario específico.

Si no tenemos resultado alguno, el nuevo nombre de usuario no está ya en uso y se puede insertar en la tabla usuario todos los datos:

```
insert into Usuario values ('login','passCodificada',
 'nombre', 'apellidos','titulo', 'despacho', 'mail',
  'tfno1','tfno2', 'fax', 'firma', 'administrador');
```

Figura 4.4: Creación de un nuevo usuario

4.2.3.- Creación de un nuevo evento:

Para guardar un nuevo evento en la base de datos, es necesario unos ciertos pasos que incluyen alguna consulta a la base de datos:

1. Tomamos el id del evento anterior y le sumamos 1 para obtener el índice de Nuevo evento.

```
select max(IDEVENTO) from Evento;
```

Figura 4.5: Consulta para obtener el id del último evento creado.

2. Creamos el evento en la tabla EVENTO

```
  insert into Evento values( ‘nombre ’, idEvento,
‘descripcion’, ‘lugar’,tipo, recordatorio,’mensaje’,
    actualizacion,’loginUsuario’, ‘confirmado’,
                ‘cancelado’);
```

Figura 4.6: Sentencia para la creación de un nuevo evento

3. Introducción de invitados:

Se recorre cada uno de los invitados al evento y se utiliza la siguiente inserción.

```
    insert into EsInvitado values( ‘emailInvitado’,
            idEvento,’comentario’);
```

Figura 4.7: Sentencia para almacenar los invitados respecto a un evento creado

Naturalmente, como el comentario se rellenará si un invitado quiere aportar algo, éste se crea vacío.

4.2.4. Inserción de la/s fecha/s.

A la hora de la inserción de fechas nos encontramos con dos posibles casos: que el evento sea confirmado o no confirmado.

a.- Si el evento es confirmado, sólo se podrá introducir una fecha para ese evento, y tanto el evento como ésta llevarán en su propiedad CONFIRMADO=1.
b.- Por otro lado, si no es confirmado, los dos tendrán CONFIRMADO=0.

Tomamos primeramente el id de la última fecha creada, y le sumamos 1 como hicimos con Evento para obtener el id correspondiente a la nueva fecha:

```
select max(IDEVENTO) from Fecha;
```

Figura 4.8: Consulta para obtener el id de la ultima fecha creada

Y después, tomamos todos los datos de la fecha y lo guardamos.

```
insert into Fecha values (IDFECHA,'FECHAVALUE',
     CONFIRMADO, 'LOGINUSUARIO', IDEVENTO,
          'DURACION', NVOTOS);
```

Figura 4.9: Sentencia para la creación de una nueva fecha relacionada a un evento ya creado

Lógicamente, al ser una nueva fecha, el valor del número de votos (NVOTOS) es 0 hasta que alguien la vote.

Por ultimo, se rellena la asociación de fechas con las respuestas de los invitados mediante:

```
insert into VotaFechas values ('EMAILINVITADO',
            IDFECHA, 'RESPUESTA');
```

Figura 4.10: Sentencia para almacenar la relación de las repuestas de los invitados respecto a una fecha

4.2.5.- Un invitado vota una fecha:

Una vez que un invitado expresa su respuesta, se modifica el registro con ésta así como el número de votos de la fecha.

Primero obtenemos el número de votos asociado a esa fecha. Solamente se suma un voto si su respuesta es “SI”; si la respuesta es “Posiblemente” o “No”, no se suma nada. Así mismo, si esa fecha ya estaba votada pero por cualquier motivo se modifica, el número de votos sufre también la modificación.

```
select NVOTOS from Fecha where IDFECHA= 'IDFECHA’;
  update Fecha set NVOTOS = VOTOS where IDFECHA =
                    IDFECHA;

   update VotaFechas set RESPUESTA= 'respuesta '
where EMAILINVITADO = 'EmailInvitado' and IDFECHA =
                    idFecha;
```

Figura 4.11: Sentencias para actualizar las votaciones de cada fecha

4.2.6.- Un usuario consulta sus eventos:

Un usuario registrado puede acceder a la aplicación y ver el evento que el mismo ha creado o a los que ha sido invitado. Los eventos se muestran por orden de más nuevo a más antiguo:

- Eventos creados:

```
select * from Evento where LOGINUSUARIO='login'
            order by IDEVENTO desc;
```

Figura 4.12: Sentencias para obtener los eventos creados por un usuario

- Eventos a los que se es invitado:

```
        select * from Evento, EsInvitado
  where (EsInvitado.EMAILINVITADO='emailUsuario'
and EsInvitado.IDEVENTO=Evento.IDEVENTO) order by
              Evento.IDEVENTO desc;
```

Figura 4.13: Sentencias para obtener los eventos a los que ha sido invitado un usuario

4.2.7.- Actualizar la información personal:

Esta es la sentencia básica para modificar los campos de un usuario. Se pueden ir completando así si los campos no obligatorios son rellenados o no.

```
update Usuario set (campos a modificar)
        where login='login';
```

Figura 4.14: Sentencias para actualizar los campos que se requieran de la información de un usuario

La sentencia completa con todos los campos sería la siguiente:

```
update Usuario set PASSWORD ='password',
NOMBRE='nombre', APELLIDOS='apellidos', TITULO='titulo',
        DESPACHO='despacho', EMAIL='email',
    TELEFONO1='telefono1', TELEFONO2='telefono2',
  ,FAX='fax', FIRMA= 'firma', ADMINISTRADOR= 'admor'
        where login='login';
```

Figura 4.15: Sentencia modelo con todos los posibles campos a actualizar de la información de un usuario

4.2.8.- Un usuario confirma una fecha:

Cuando un usuario confirma la fecha de un evento, interiormente se hace que ese evento tenga el campo CONFIRMADO a SI y la propiedad CONFIRMADO de Fecha a 1

```
update Evento set CONFIRMADO = 'SI'
    where IDEVENTO = idEvento;

update Fecha set CONFIRMADO = 1 where IDFECHA = idFecha;
```

Figura 4.16: Sentencias para actualizar los campos que se requieran de la información de un usuario

4.2.9.- Excluir algún invitado del evento:

Es necesario cumplir un orden de borrado sobre las tablas en el cual primero se borran los registros del invitado que no tienen asociados otros de la base de datos.

```
update Fecha set Fecha.NVOTOS = (Fecha.NVOTOS - 1)
     where (Fecha.IDFECHA =idFechaVotadaSI);

          delete FROM EsInvitado WHERE
EMAILINVITADO='emailInvitado' and IDEVENTO=idEvento;

        delete from VotaFechas, Fecha where
 (VotaFechas.EMAILINVITADO='emailInvitado' and and
VotaFechas.IDFECHA=(select Fecha.IDFECHA from Fecha
        where  Fecha.IDEVENTO=idEvento));
```

Figura 4.17: Sentencias para quitar a alguien de los invitados

4.3 Opciones de administración:

4.3.1.- Dar de alta un nuevo usuario:

Antes de dar de alta un usuario, es necesario comprobar que el nombre de usuario introducido es único:

```
SELECT * FROM Usuario where LOGIN='login';
```

Figura 4.18: Consulta para obtener todos los usuarios con un nombre de usuario específico.

Si el resultado es -1, significa que no hay nadie con ese nombre de usuario y se puede crear un nuevo usuario.

```
insert into Usuario values ('login', ’passcodificada’,
'nombre', 'apellidos', 'titulo', 'despacho’, 'email’,
      'telefono1', 'telefono2’, ’fax’, 'firma',
                  'adminsitrador');
```

Figura 4.19: Consulta para crear un nuevo usuario

En caso contrario, se puede consultar todos los usuarios donde se ven los nombres de usuario dados, y dar de baja a alguno si procede.

4.3.2.- Dar de baja un usuario:

Dar de baja un usuario requiere de unas complejas sentencias SQL para poder borrar de la Base de Datos todos los registros asociados al usuario a dar de baja como son los eventos **creados** y las fechas. Para ello, es necesario empezar borrando las referencias a las tablas que no apuntan a otras, y así sucesivamente.

```
delete EsInvitado from EsInvitado, Evento, Usuario
 where ( EsInvitado.IDEVENTO=Evento.IDEVENTO and
      Evento.LOGINUSUARIO=Usuario.LOGIN and
         Usuario.LOGIN='loginUsuario');

 delete VotaFechas from VotaFechas,Fecha, Evento,
                  Usuario where
      (VotaFechas.IDFECHA=Fecha.IDFECHA and
       Fecha.IDEVENTO=Evento.IDEVENTO and
      Evento.LOGINUSUARIO=Usuario.LOGIN and
         Usuario.LOGIN='loginUsuario');

  delete Fecha from Fecha, Evento, Usuario where
       (Fecha.IDEVENTO=Evento.IDEVENTO and
       Evento.LOGINUSUARIO=Usuario.LOGIN and
         Usuario.LOGIN=' loginUsuario ');

    delete Evento from Evento, Usuario where
     (Evento.LOGINUSUARIO=Usuario.LOGIN and
        Usuario.LOGIN=' loginUsuario ');

delete from Usuario where LOGIN=' loginUsuario ';
```

Figura 4.20: Sentencias necesarias para dar de baja un usuario de la aplicación

Capitulo 5

Componentes de la aplicación

El diseño de Rendez-Vous sigue la arquitectura Modelo-Vista-Controlador planteada por Java. Tal y como se detalla en **java.sun.com** ***[4]***, el problema que se suele tener a la hora de diseñar una aplicación es que ésta debe poder soportar distintos tipos de usuarios con distintas interfaces.

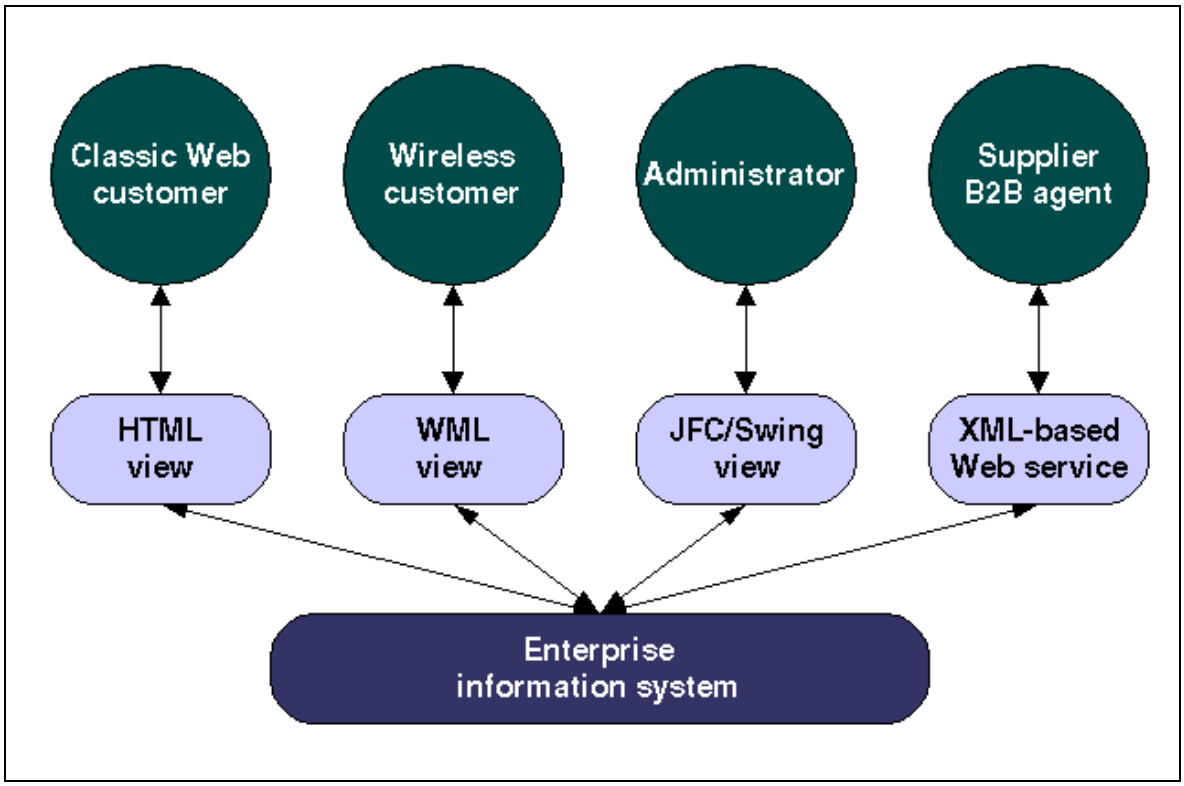

Figura 5.1: Problemática de distintos entornos de interfaz que deben de soportar las aplicaciones

Aunque por el momento aquí solamente se soportará que los usuarios que accedan por Web, se deja el sistema preparado por si en alguna futura ampliación se decidiese hacer un cliente para usos locales de usuarios o así como acceso por dispositivo móvil.

La aplicación de la arquitectura Modelo-Vista-Controlador (MVC) a una plataforma JAVA hace que se separe los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Esta separación permite múltiples vistas para la compartición de los mismo sistemas de gestión de datos, lo que hace facilita la implementación de múltiples clientes, su pruebas y mantenimiento.

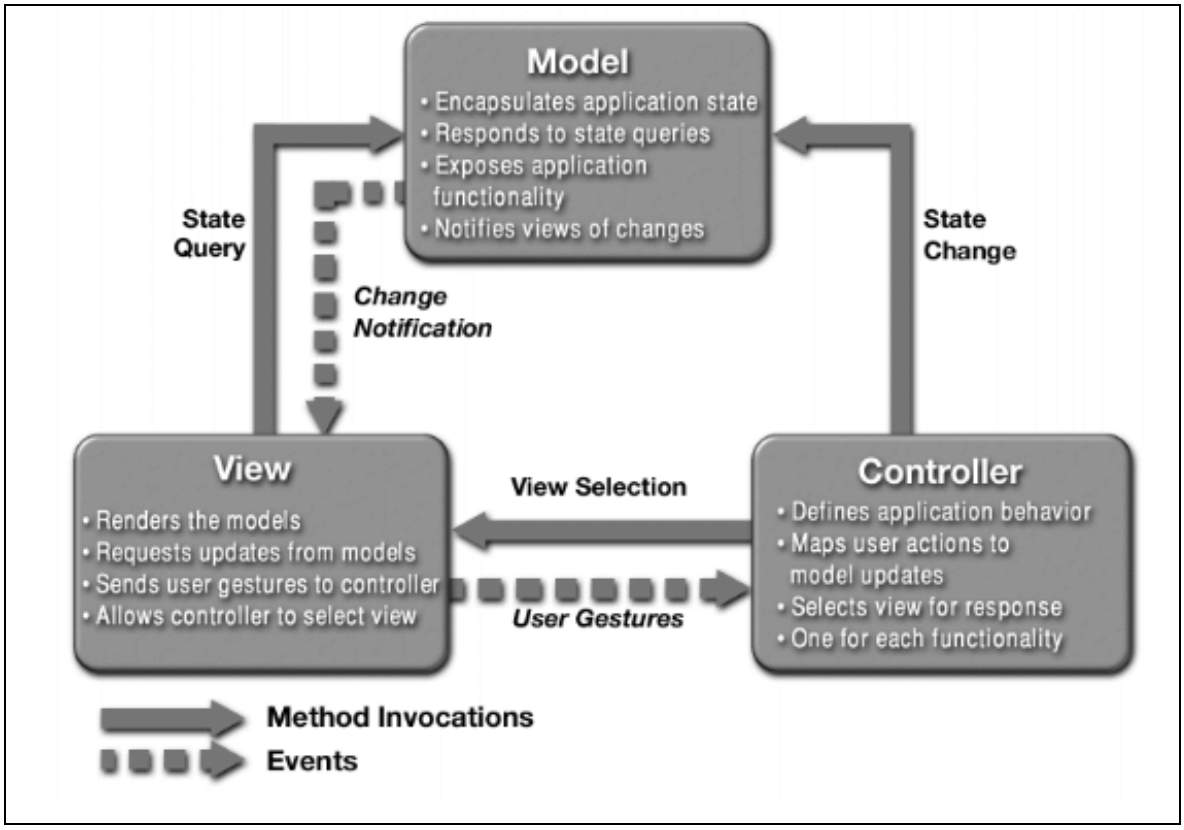

Figura 5.2: Arquitectura Modelo-Vista-Controlador

Modelo: El modelo representa los datos y las reglas que gobiernan el acceso y las actualizaciones de estos datos. Es la representación específica de la información con la cual el sistema opera y con su lógica de datos asegura la integridad de éstos y permite derivar nuevos datos.

Vista: La Vista presta el contenido al modelo. Accede a los datos a través del modelo y especifica como deberían representarse los datos. La responsabilidad de la vista es mantener la consistencia en su presentación cuando el modelo cambia, lo que puede ser conseguido usando un “push model”, donde la vista notifica a ella misma las notificaciones de los cambios, o un “pull model” donde la vista es responsable de llamar al modelo cada vez que necesite obtener los datos mas actuales. Además, la Vista presenta el modelo en un formato adecuado para interactuar, generalmente la interfaz de usuario. En este caso, correspondería a los JSP.

Controlador: El Controlador transforma las interacciones con la vista en acciones que deben llevarse a cabo por el Modelo. En una interfaz gráfica de un cliente, las interacciones del usuario pueden ser *clicks* de botones o las selecciones de un menú, mientras que en una aplicación Web, éstas aparecen como GET y POST de peticiones HTTP. Las acciones hechas por el modelo incluyen activar los procesos de negociación o cambiar el estado del modelo. Basado en las interacciones del usuario y en la salida del modelo, el Controlador responde seleccionando la Vista apropiada. Esto correspondería a los Servlets.

5.1 MODELO:

El modelo de la aplicación *Rendez-Vous* que nos ocupa, se compone de siete clases que son:

1.- Codificar.java
2.- Conexión.java
3.- Consulta.java
4.- Email.java
5.- EnviarCambio.java
6.- EnviarInvitacion.java
7.- EnviarPassword.java

Se va a proceder a definir cada una de ellas.

5.1.1 Codificar.java

Esta clase se usa exclusivamente para codificar un String que le pasen en otro usando un **HASH** y la codificación **SHA-1**, que proporciona un gran nivel de seguridad de codificación. Si se descifrase un 90% de una clave codificada con SHA-1, esto no sería suficiente para descifrar la clave entera. Para ello, cuenta con un único método que recibe el `String` a codificar y devuelve otro codificado a partir de este:

```
public synchronized String encrypt(String plaintext) {
    MessageDigest md = null;
    String hash=null;
    try
    {
      md = MessageDigest.getInstance("SHA-1"); //Paso 1
      md.update(plaintext.getBytes("UTF-8")); // Paso 2
      byte raw[] = md.digest(); //Paso 3
      String hash = (new BASE64Encoder()).encode(raw); //Paso4
    }
    catch(NoSuchAlgorithmException e)
    {
       System.err.print("Fallo al crear algoritmo ” +
                        “de cifrado paso 1");
    }
    catch(UnsupportedEncodingException e)
    {
      System.err.print("Fallo al crear algoritmo “ +
                        ”de cifrado paso 2");
     }
    return hash; //Paso 5
  }
```

Figura 5.3: Método de encriptación mediante Hash

Primero, se pide al API de seguridad de Java crear una nueva instancia de un objeto `MessageDigest`**,** definiendo el algoritmo de cifrado. En este caso, se usa SHA-1.

A continuación hay que convertir la variable de tipo string que se ha de codificar, en su equivalente en formato binario. Para este paso nos valemos del formato de codificación de caracteres UTF-8. Utilizamos el resultado obtenido con el objeto `MessageDigest` que hemos creamos anteriormente para de ese modo crear el array que finalmente contendrá la contraseña ya codificada.

Finalmente se lleva a cabo las últimas operaciones de codificación, como el *padding* (que sirve para indicar la longitud del mensajes y comprobar si es correcto), apoyándose para ello en las funciones y metodologías de encriptación *hash*. Tras ello ya se está en condiciones de crear en un array el resultado con la contraseña codificada que por último se pasa a formato Base64, devolviendo el array codificado.

A continuación se puede observar un ejemplo de una url incluida en un correo para invitar a un usuario a una reunión con su clave codificada en SHA-1 para hacerla única.

```
http://rendezvous.com/pfc/VotarReunion?idEvento=36&invitado=
marta@dominio.com&key=APq1tHWYZEqfTqvvY9QMpOucQ%2Bw=
```

Figura 5.4: Ejemplo de una URL incluida en una invitación

5.1.2 Conexion.java

Gracias a esta clase, es posible establecer la conexión contra la base de datos de MySQL usando para ello la tecnología conocida como JDBC.

Para ello, es necesario que primero especifiquemos el tipo driver que se usará para establecer la conexión. Tras este paso indicaremos los datos necesarios para establecer la conexión propiamente dicha.

```
protected Connection con;
    private String driver = "com.mysql.jdbc.Driver";
    private String user = "marta";
    private String password = "*****";
    private String url = "jdbc:mysql://localhost:3306/mw";

public Conexion() {
    try{
      Class.forName(driver).newInstance();
     con = DriverManager.getConnection(url, user,
                                        password);
     } catch (SQLException sqle) {
            System.out.println(sqle);
            System.exit(1);
       }
}
```

Figura 5.5: Método de conexión con una base de datos

5.1.3 Consulta.java

Consulta es la encargada de ejecutar todas las sentencias SQL necesarias en la aplicación sobre la base de datos. Para ello, se apoya sobre un objeto conexión para acceder a ella. Por otra parte, es necesario también el uso de un objeto `Statement` que recoge la sentencia a ejecutar y un `ResultSet`, donde se almacena el resultado de esta ejecución.

En esta clase, a pesar de que se pueden usar mas tipos de métodos de `Statement`, sólo se han usado `executeQuery(String sql)` y `executeUptade(String sql)`.

- `executeQuery(String sql)` es usado cuando se quiere obtener algún dato de la Base de Datos. Se devolverá un objeto `ResultSet` donde hay que recorrer uno a uno sus elementos para obtener la información solicitada.

```
sentencia = con.createStatement();
ResultSet answ = sentencia.executeQuery("SELECT * FROM”+
                        “Evento WHERE IDEVENTO="+id+ ";");
if(answ.next()){
   do{
     evento.setNombre(answ.getString(1));
     evento.setIdEvento(answ.getInt(2));
     ...
     }while(answ.next());
 }
```

Figura 5.6: Método para ejecutar una consulta sobre una base de datos

- `executeUpdate(String sql)` se usa cuando se quiere introducir nuevos datos, así como hacer alguna actualización sobre la base de datos.

```
try{
   sentencia = con.createStatement();
   String consulta="update Evento set CONFIRMADO = 'SI' ”+
                   “where IDEVENTO = " +
                   ev.getIdEvento()+ “;";
   int actualizacion = sentencia.executeUpdate(consulta);
 }catch(Exception e12){
     System.err.println(e12);
     return -1;
 }
```

Figura 5.7: Método para ejecutar una actualización sobre una base de datos

5.1.4 Email.java

La clase Email.java, utilizando el API JavaMail, es la encargada de establecer una conexión con un servidor SMTP, autenticarse si es necesario y enviar cualquier notificación vía email a una determinada dirección de correo electrónico.

El método que se encarga de invocar todos los parámetros para poder proceder a realizar las acciones descritas anteriormente, es el método `enviar()`.

```
public void enviar(String to, String subject, String msj) {

    //Creamos el correo
    this.to = to;
    this.subject = subject;
    this.mensaje = msj;

    //Hacemos todo lo necesario para enviarlo
    Properties prop = new Properties();
    prop.put("mail.smtp.host", host);

    /*Esta línea es la que indica al API que debe autenticarse*/
    prop.put("mail.smtp.auth", "true");

    try {
         SMTPAuthentication auth = new SMTPAuthentication();
         Session session = Session.getInstance(prop, auth);

         Message msg = getMessage(session, mensaje);
         Transport.send(msg);
    } catch (Exception e) {
         ExceptionManager.ManageException(e);
    }

}
```

Figura 5.8: Método para el envío de un mensaje vía SMTP

Para ello, y tras haber importado los paquetes necesarios es necesario sobrescribir las propiedades de JavaMail con los parámetros necesarios para una configuración propia.

Como se puede observar en el código, se crea un objeto `Properties` indicando el nombre o IP del servidor SMTP que vamos a utilizar, e indicando en el segundo parámetro si es necesario autenticarse o no.

Si el servidor SMTP contra el que nos conectamos necesita autenticación, es necesario además del código previo, invocar un objeto de la siguiente clase que permite llevarlo a cabo:

```
class SMTPAuthentication extends javax.mail.Authenticator {

    public PasswordAuthentication getPasswordAuthentication()
{
     String username = "marta@dominio.com";
     String password = "********";

     return new PasswordAuthentication(username, password);
    }
}
```

Figura 5.9: Método para la autenticación con un servidor SMTP

Con los objetos `Properties` y `SMTPAuthentication` se crea un objeto `Session` a través del método estático `getInstance()` de la propia clase `Session`. La clase `Session` pertenece al API y define la sesión utilizada para establecer la conexión con el servidor.

Para una mejor organización, se ha aislado la formación del mensaje en el método `getMessage()`, que devuelve un objeto `MimeMessage`, también definido por el API, y que representa el mensaje a enviar.

```
private MimeMessage getMessage(Session session, String mensaje) {
try {

MimeMessage msg = new MimeMessage(session);

msg.addRecipients(Message.RecipientType.TO, to);

msg.setHeader("Content-Type","text/plain; charset=iso-8859-1");
msg.setHeader("Content-Transfer-Encoding", "quoted-printable");

msg.setFrom(new InternetAddress(from, "Rendez-Vous"));

msg.setSubject(subject);
msg.setText(mensaje);

return msg;

} catch (java.io.UnsupportedEncodingException ex) {
ExceptionManager.ManageException(ex);
return null;

} catch (MessagingException ex) {
ExceptionManager.ManageException(ex);
return null;
}

}
```

Figura 5.10: Método para la creación de mensajes de tipo MIME

Con este método lo que se pretende es definir la estructura tanto de la cabecera como del cuerpo del Email.

Primero, se crea una nueva instancia de un mensaje. Seguidamente, se procede a definir tanto los receptores de este mensaje. En este caso, se crea un mensaje por receptor de mensaje, y para ello se apoya en destinatario definido en el método `enviar()` y en el atributo `to` de la clase Email.java.

A continuación, definimos las propiedades del mensaje. Se empieza definiendo las distintas cabeceras que se tienen. En estas cabeceras, se también la codificación que se usará, en este caso "***ISO-8859-1***", que corresponde a los caracteres de Europa occidental.

Luego se pone al mensaje el asunto que tendrá y el contenido del cuerpo del mensaje, y se devuelve este mensaje si todo ha salido bien.

Por último, enviamos el mensaje a través del método estático `send()` de la clase `Transport`. La clase `Transport` es la clase encargada de la entrega de mensajes al servidor.

Si se produjese algún tipo de problema o excepción, la encargada de avisar con un mensaje por pantalla es la clase `ExceptionManager`.

```
class ExceptionManager {
    public static void ManageException(Exception e) {
        System.out.println("Se ha producido “ +
                           ”una exception");
        System.out.println(e.getMessage());
        e.printStackTrace(System.out);

    }
}
```

Figura 5.11: Excepción si algo sale mal al crear un mensaje

5.1.5 EnviarCambio.java

Esta clase, apoyada en ***Email.java***, es utilizada para notificar a los distintos invitados, cualquier modificación que haya dentro de los detalles del evento.

5.1.6 EnviarInvitacion.java

Esta clase también en ***Email.java***, se usa para notificar a los distintos invitados de todas y cada una de las reuniones a las que hayan sido convocados. Esto se consigue mediante el envío de una URL única con una clave, evitando de ese modo posibles suplantaciones de identidad. A su vez se les remite a un 'servlet' donde podrán fijar sus disponibilidades frente a un evento al que hayan sido invitados.

5.1.7 EnviarPassword.java

Si en cualquier momento es necesario restablecer la password de un usuario, mediante esta clase, y como en las anteriores apoyándose en Email.java, se establecerá una contraseña por defecto y se enviará un correo electrónico notificándolo al usuario.

5.2 VISTA y CONTROLADOR

Como se ha explicado anteriormente, el cometido de la Vista de la aplicación es mostrar los datos al usuario y recogerlos para pasárselos, a través del Controlador, a las clases Modelo que se han visto en el apartado anterior.

Por otra parte el controlador es el encargado de ejecutar esas interacciones entre la Vista y el Modelo.

A continuación, se va a proceder a detallar las principales casuísticas de la aplicación para resaltar las clases que intervienen tanto en la Vista como en el Controlador:

5.2.1.- Autenticación de un usuario:

Nada mas cargar la aplicación, lo primero que sucederá es que se pida al usuario, si este no lo ha hecho anteriormente, que se autentique con su usuario y contraseña.

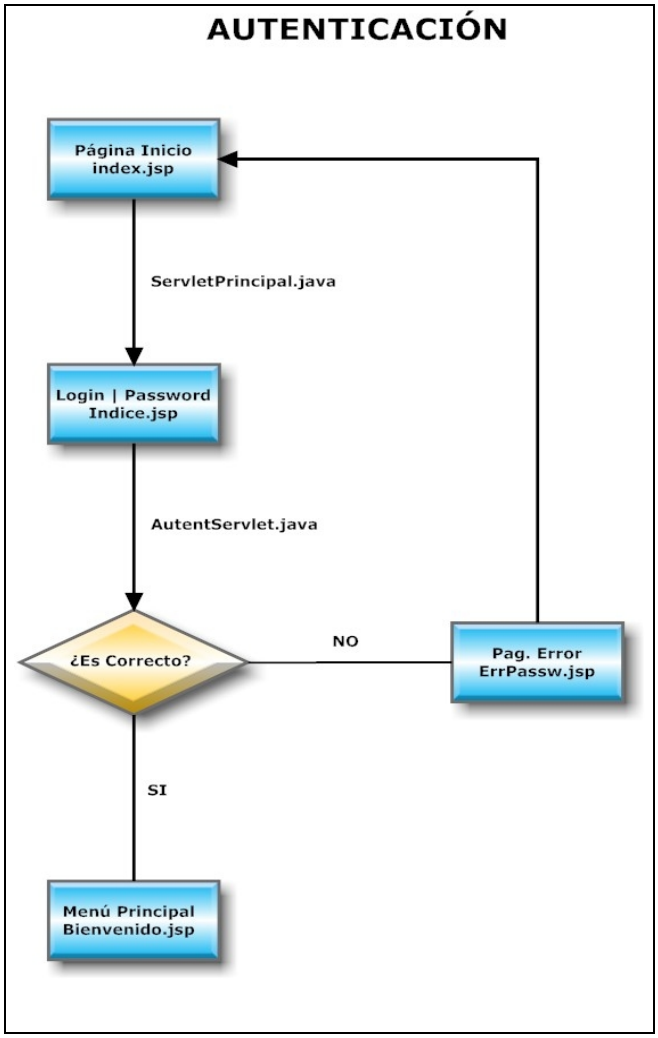

Figura 5.12: Diagrama de flujo del proceso de autenticación

Como se puede observar, si a la hora de realizar la consulta de autenticación se devuelve un valor negativo, se redirige al usuario a una página de error, mientras que si todo está bien, se le redirige al menú principal.

5.2.2.- Elegir una opción en el menú principal:

Una vez que el usuario se ha autenticado, se le ofrecen 3 opciones en las que puede elegir o crear una nueva reunión, ver las reuniones creadas o invitadas y cambiar la información personal:

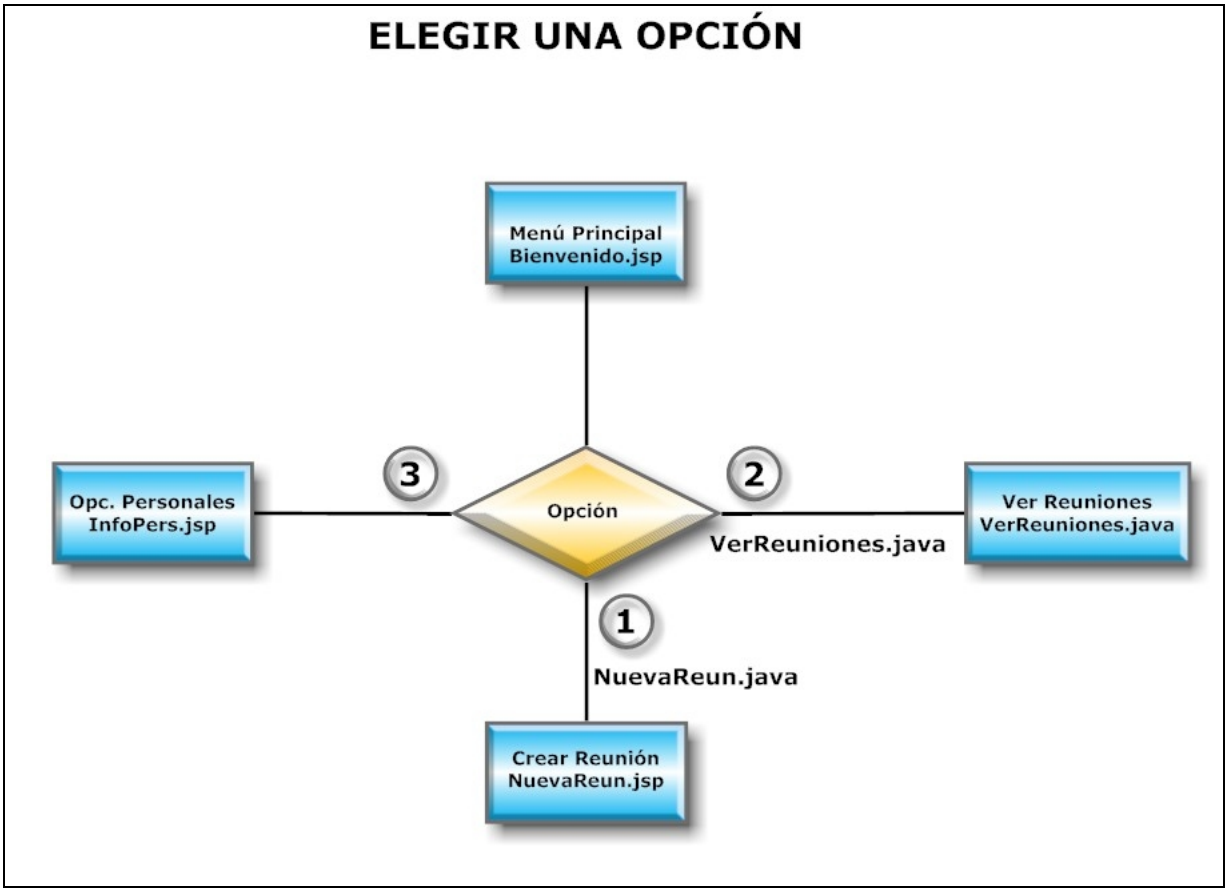

Figura 5.13: Diagrama de flujo del menú principal

5.2.2.1.- Creación de una nueva reunión:

Dentro de esta opción se puede elegir si la reunión a crear es con fecha confirmada o si está abierta a varias fechas, esperando a ver las respuestas de los invitados y decidiendo en función de ellas la mejor fecha para reunirse.

Internamente, los jsp y servlets utilizados son iguales salvo el de elección de fecha, en el que el primer caso solamente permite elegir una fecha y en el segundo esta abierto a recoger tantas fechas como quieran ser introducidas.

Figura 5.14: Diagrama de flujo de la creación de una reunión

5.2.2.2.- Ver Reuniones:

Aquí los usuarios pueden revisar las reuniones que han organizado así como a las que han sido invitados. Además, pueden obtener un fichero .ics con las reuniones pendientes ya confirmadas que ha organizado así como a las que han sido invitados y han confirmado su asistencia.

Un usuario que ha organizado un evento, puede en una primera instancia ver sus detalles o cancelarlo. Si decide cancelarlo, dispondrá de una opción para avisar a los invitados.

Por otra parte, una vez que consulte los detalles de la reunión, se encontrará con una amplia variedad de opciones para gestionarla adecuadamente. En esta opción, las principales acciones que se pueden hacer es confirmar o añadir una fecha, si estamos con un evento aun no confirmado. Además, se puede modificar los detalles de la reunión así como enviar un recordatorio a todo el mundo o a los invitados que aun no hayan fijado sus preferencias.

Por otra parte, en este mismo apartado se puede ver de una forma más personalizada las opciones de cada invitado, pudiendo contestar por él, mandarle un recordatorio personalizado o quitarle de la reunión.

Si se consulta una reunión a la que ha sido invitado, el usuario no encontrará las opciones que hemos mencionado antes, al no tratarse de un organizador del evento.

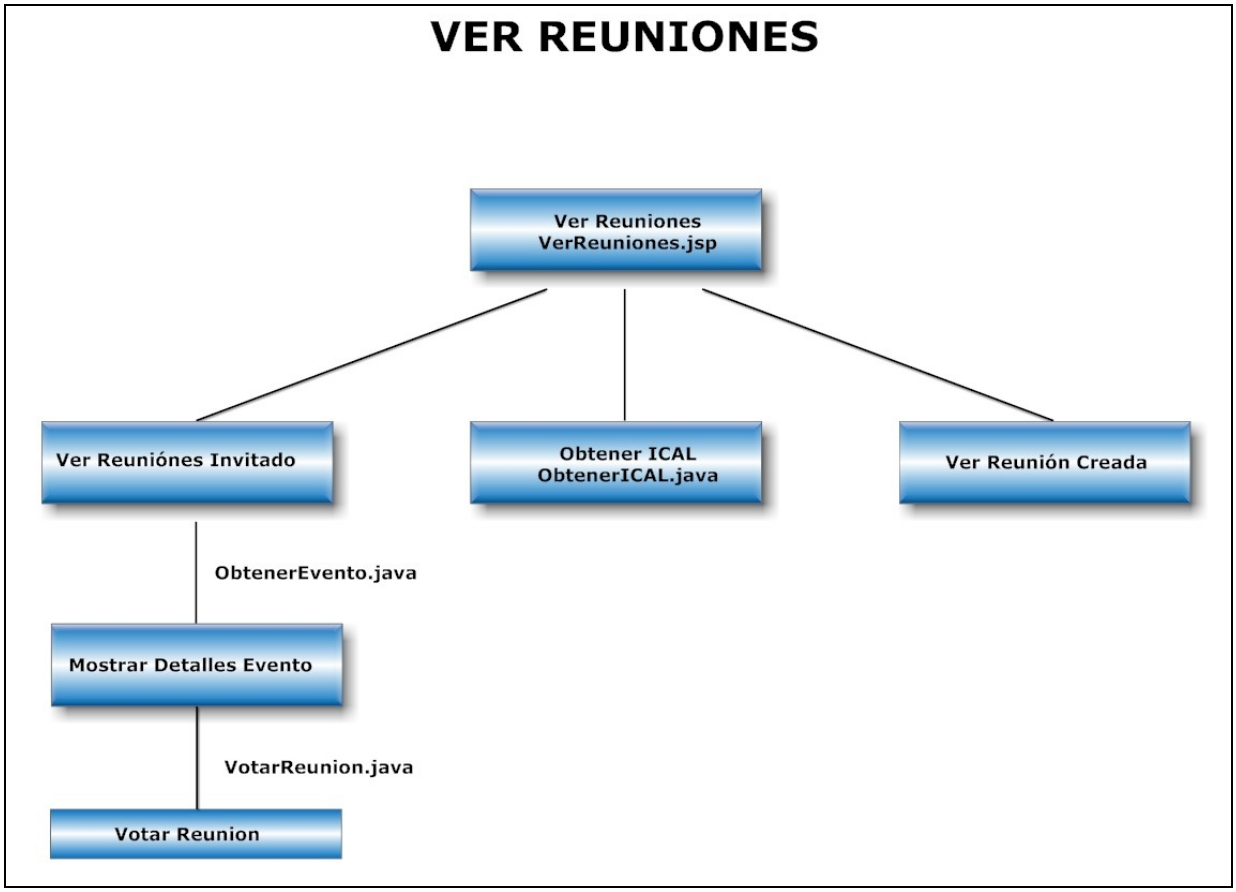

Figura 5.15: Diagrama de flujo que permite consultar las reuniones pendientes

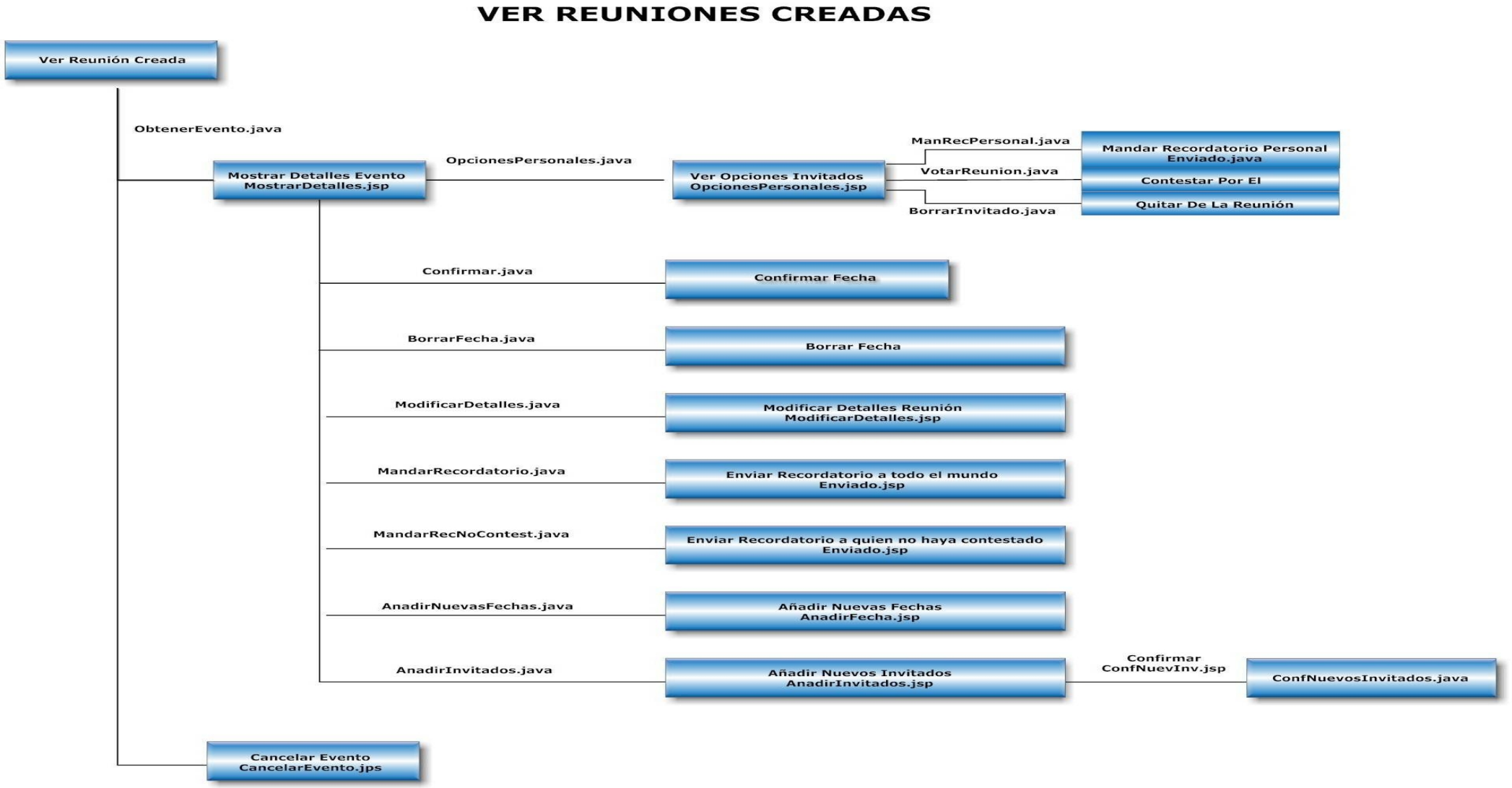

Figura 5.16: Diagrama que muestra las opciones para la visualización de reuniones creadas

5.2.2.3.- Cambiar la información personal:

Rendez-Vous permite a los usuarios que puedan cambiar su información personal.

A parte de cambiar su información personal, los usuarios pueden cambiar la cuenta de correo electrónico que tiene de referencia de Rendez-Vous.

Por último, cualquier usuario puede cambiar su contraseña de acceso. Es recomendable que el usuario la cambie la primera vez que accede a la aplicación debido a que a la hora de la creación de un usuario se genera una contraseña genérica.

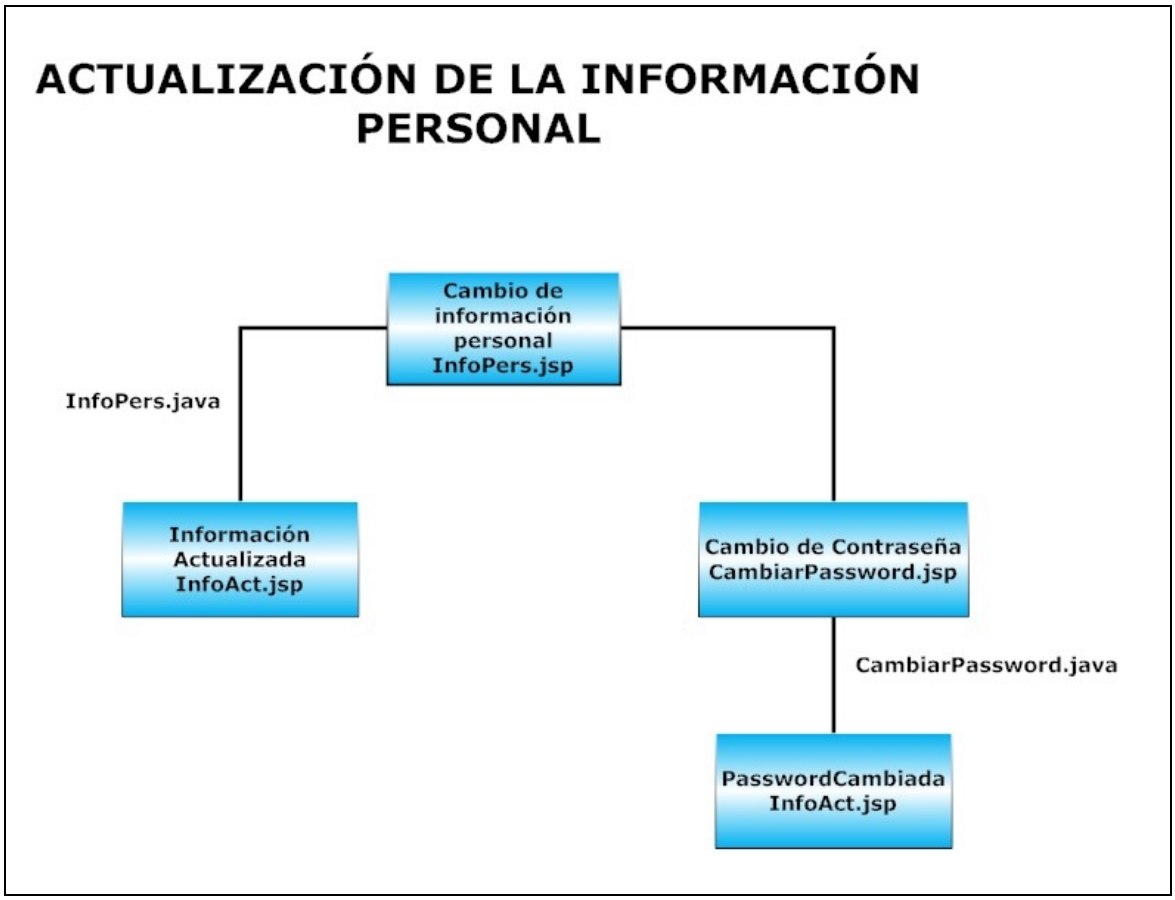

Figura 5.17: Diagrama de flujo que permite cambiar la información personal

5.2.3 Administración

La aplicación Rendez-Vous cuenta con una opción visible solo a los usuarios administradores que sirve para administrar las altas y las bajas de usuarios. Para ello, cuenta con una interfaz muy sencilla de usar que permite crear un usuario, comprobando que el usuario asignado a este no este ya en uso. Si por casualidad lo estuviera, se provee una opción donde se puede consultar quien lo tiene y decidir si su antiguo dueño lo sigue necesitando o no.

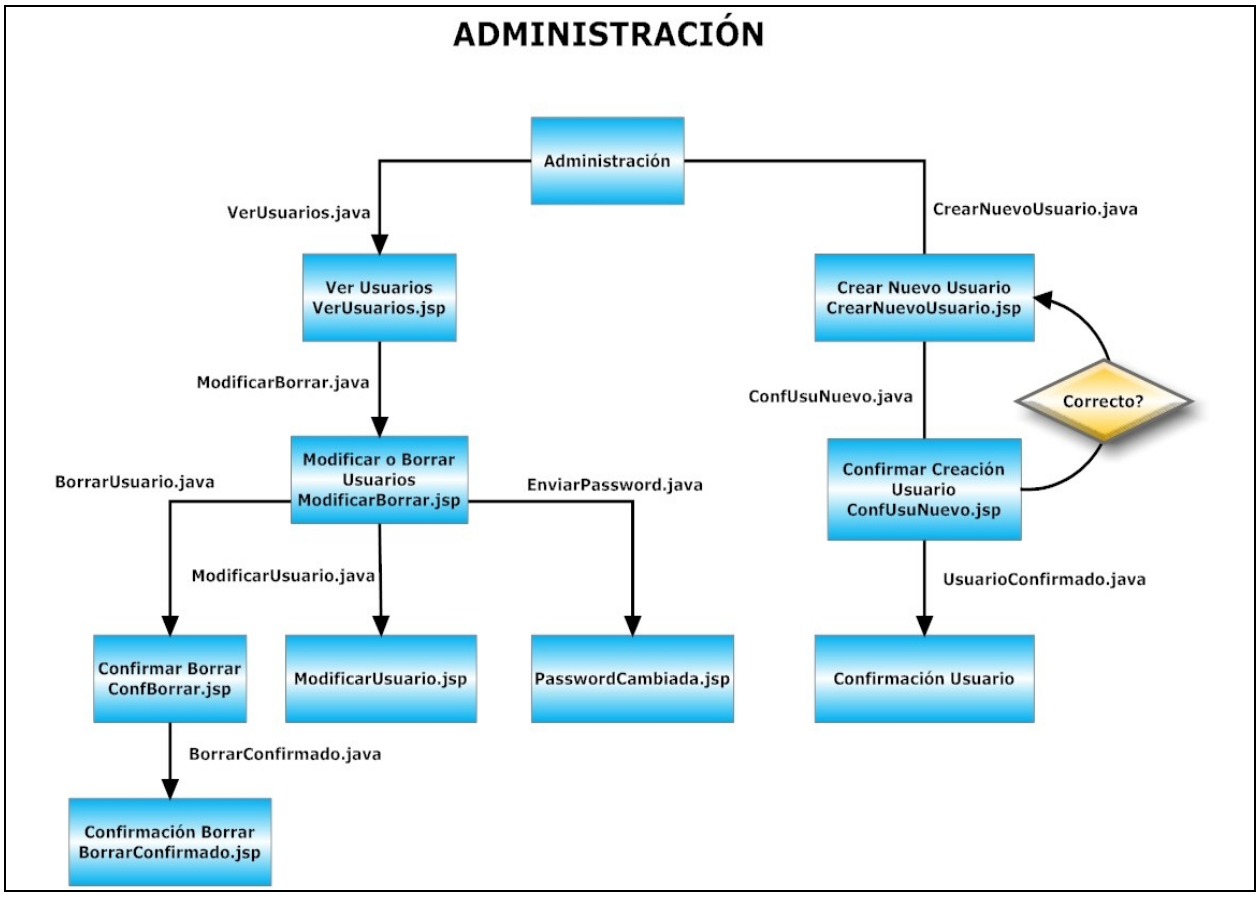

Figura 5.18: Diagrama de flujo de la parte administrativa

Capítulo 6

Pruebas, Conclusiones y Trabajos Futuros

6.1 Pruebas

Para el entorno de pruebas se ha instalado la aplicación sobre un sistema Ubuntu versión 8.04 y la versión de Java utilizada ha sido la 1.5.0_14. Así mismo, la versión del servidor Apache-Tomcat utilizada es la 5.5.12.

La máquina utilizada para la ejecución de las pruebas es un servidor Pentium IV HT 2.80 GHz y 2 GB de memoria RAM. La conexión de red externa que tiene el servidor es una línea ADSL cuyo bitrate de bajada son 8Mbps y de subida 640 Kbps.

Las pruebas realizadas a la aplicación se dividen por un lado en la consistencia a la hora de almacenar toda la información en la base de datos y por otro lado en la robustez de la aplicación Web en sí a la hora de recoger la información introducida por los usuarios y mostrarla. Finalmente, se procedió a simular los escenarios más típicos que se pueden dar en la aplicación y a observar tanto los tiempos de respuesta y conexiones a la aplicación desde cinco, diez y veinte clientes simultáneos teniendo siempre en cuenta las características del entorno que se dispone. Para esta ultima parte, se ha utilizado el programa *Microsoft Web Application Stress [21].*

6.1.1 Recuperación y almacenamiento de datos:

Una vez diseñadas las diferentes consultas a ejecutar sobre la base de datos MySQL, y tras muchas pruebas, se observó que en las tablas implicadas se introducían correctamente los valores necesarios y que igualmente se creaban las relaciones entre las tablas, tal y como se especificó en el diseño de la base de datos.

Además, MySQL ofrece un control integrado de la concurrencia en el acceso y modificación de los datos almacenados en las tablas.

6.1.2 Aplicación Web:

La función más importante que se desempeña en este bloque es la comprobación de los datos que se muestran en el navegador así como la de los datos que son introducidos por un usuario. Se verifica que estos datos cumplen con los formatos establecidos, y que ningún campo requerido se deja en blanco. Siempre que se detecta que alguno de los datos introducidos es erróneo, la aplicación redirecciona al usuario a una pantalla dónde se le solicitará que compruebe los datos introducidos, y lo intente de nuevo.

Por otra parte, se detectó que la sesión caducaba cuando pasaban un par de minutos sin actividad. Para permitir que la sesión tuviera un mayor margen de tiempo de inactividad y evitar que las sesiones caducasen con demasiada frecuencia, se decidió establecer el tiempo máximo de una sesión inactiva a 20 minutos. De este modo era suficiente para que no caducaran las sesiones demasiado a menudo, aportando también que no se quedasen abiertas de forma permanente. Para ello se empleó el parámetro:

```
<session-config>
        <session-timeout>
            20
        </session-timeout>
</session-config>
```

en el fichero web.xml. Además y como medida de seguridad para que ningún usuario pueda acceder a la aplicación sin autenticarse, se añadió que la sesión se crease solamente en el momento de la autenticación al inicio del programa. En cada paso, se vuelve a comprobar si esa sesión todavía existe, estableciendo que no se cree otra vez.

Si se da el caso en el que la sesión ha expirado lo que se hace es redireccionar a una página donde se informa de ello y se le pide al usuario que se vuelva a autenticar creando de este modo una nueva sesión.
Además de todo lo anterior la aplicación Web cumple los estándares de XHTML haciendo que sea portable entre distintos navegadores como pueden ser Firefox, Internet Explorer, Opera o Safari.

Respecto al envío por SMTP mediante Javamail, lo que se hizo primero fue desarrollar un programa que pudiera simplemente conectarse con un servidor SMTP y enviar un correo electrónico. Tras probarlo y ver que los resultados eran buenos, se incorporó a la aplicación para poder notificar las distintas invitaciones y cambios de una reunión.

6.1.3 Pruebas de estrés

Para las pruebas de estrés se han planteado 3 escenarios a testear: uno de carga baja para el sistema como es la autenticación, uno de carga media como es la obtención de todas las reuniones pendientes y uno de carga alta como es la creación de una reunión con su correspondiente envío de notificaciones a los invitados y almacenamiento en la base de datos.

Cada uno de los escenarios anteriores se ha sometido a la prueba estrés con cinco, diez y veinte clientes accediendo constantemente durante un tiempo de un minuto, sin simular el retardo de pulsaciones que se podría dar con una persona (no hay tiempos de espera entre petición y petición para forzar aun más el estrés).

6.1.3.1 Primer escenario: Autenticación de un usuario en el sistema.

Se trata del escenario más sencillo. Simplemente se entra y se sale de la aplicación. Este escenario sirve como base para tomar referencia de tiempo.

Escenario 1: 5 clientes

```
Socket Statistics
-----------------------------------------
Socket Connects:                    416
Total Bytes Sent (in KB):           254.28
Bytes Sent Rate (in KB/s):          4.24
Total Bytes Recv (in KB):           643.75
Bytes Recv Rate (in KB/s):          10.73
```

Figura 6.1: Estadísticas de los parámetros de la conexión en el escenario 1 con 5 clientes

En esta primera tabla se puede apreciar los siguientes campos:

- El número de conexiones que se abre entre el cliente y el servidor en ese minuto.
- El número total de bytes que se han enviado del cliente al servidor en un minuto.
- La velocidad media de envío en kilobytes por segundo durante ese minuto.
- El número total de bytes que se han recibido del servidor al cliente en un minuto.
- La velocidad media de recepción en kilobytes por segundo durante el minuto.

```
Page Summary
Page                                  Hits      TTFB Avg      TTLB Avg
=============================================================================
GET /pfc2                             42          118.33      278.00
GET /pfc2/                            42          146.50      146.50
GET /pfc2/css/telematica.css          42          153.38      153.38
GET /Icons/valid-xhtml11-blue         42          132.29      132.29
GET /pfc2/imagenes/telematica/        42          140.17      140.26
GET /pfc2/imagenes/telematica/        41          142.56      142.56
GET /pfc2/ServletPrincipal            41          152.98      154.02
POST /pfc2/AutenServlet               41          140.95      142.07
GET /pfc2/Logout                      41          142.98      144.10
```

Figura 6.2: Resumen de los tiempos medios de respuesta
en el escenario 1 con 5 clientes

En esta tabla se pueden observar los siguientes datos:

- La primera columna indica el tipo de petición realizada y muestra las páginas y objetos que han intervenido en el escenario.
- La segunda columna refleja el número de veces que durante un minuto ha intervenido dicho objeto.
- La tercera columna indica el tiempo medio de respuesta al inicio de la petición del correspondiente objeto.
- Finalmente la cuarta columna indica el tiempo medio de respuesta en el que el servidor termina la petición del objeto en cuestión.

Para explicar más detalladamente los datos se pone como ejemplo la primera entrada. Se ve que la petición:

```
GET /pfc2/ServletPrincipal
```

se ha ejecutado 42 veces. De esas 42 veces ha tardado un promedio de 118.33ms en empezar a servir y 278.00 ms en terminar de ejecutarse.

Escenario 1: 10 clientes

```
Socket Statistics
-----------------------------------------
Socket Connects:                    402
Total Bytes Sent (in KB):           245.71
Bytes Sent Rate (in KB/s):          4.10
Total Bytes Recv (in KB):           621.34
Bytes Recv Rate (in KB/s):          10.36
```

Figura 6.3: Estadísticas de los parámetros de la conexión
en el escenario 1 con 10 clientes

Page Summary

Page	Hits	TTFB Avg	TTLB Avg
GET /pfc2	40	123.40	280.40
GET /pfc2/	40	142.38	142.38
GET /pfc2/css/telematica.css	40	152.93	152.93
GET /Icons/valid-xhtml11-blue	40	134.30	134.35
GET /pfc2/imagenes/telematica/	40	142.90	142.90
GET /pfc2/imagenes/telematica/	40	147.63	147.63
GET /pfc2/ServletPrincipal	40	168.68	169.80
POST /pfc2/AutenServlet	40	161.45	162.50
GET /pfc2/Logout	40	153.63	154.85

Figura 6.4: Resumen de los tiempos medios de respuesta en el escenario 1 con 10 clientes

Escenario 1: 20 clientes

Socket Statistics

Socket Connects:	357
Total Bytes Sent (in KB):	216.64
Bytes Sent Rate (in KB/s):	3.61
Total Bytes Recv (in KB):	553.17
Bytes Recv Rate (in KB/s):	9.22

Figura 6.5: Estadísticas de los parámetros de la conexión en el escenario 1 con 20 clientes

Page Summary

Page	Hits	TTFB Avg	TTLB Avg
GET /pfc2	36	135.50	314.47
GET /pfc2/	36	178.03	178.03
GET /pfc2/css/telematica.css	36	181.61	181.67
GET /Icons/valid-xhtml11-blue	36	156.94	156.94
GET /pfc2/imagenes/telematica/	36	162.06	162.11
GET /pfc2/imagenes/telematica/	36	159.42	159.42
GET /pfc2/ServletPrincipal	35	170.23	171.80
POST /pfc2/AutenServlet	35	169.94	171.20
GET /pfc2/Logout	35	172.71	174.46

Figura 6.6: Resumen de los tiempos medios de respuesta en el escenario 1 con 20 clientes

6.1.3.2 Segundo escenario: Obtención de la lista de reuniones pendientes de cada usuario.

Escenario 2: 5 clientes

```
Socket Statistics
-----------------------------------------
Socket Connects:              166
Total Bytes Sent (in KB):     101.45
Bytes Sent Rate (in KB/s):    1.69
Total Bytes Recv (in KB):     1247.92
Bytes Recv Rate (in KB/s):    20.80
```

Figura 6.7: Estadísticas de los parámetros de la conexión en el escenario 2 con 5 clientes

```
Page Summary
Page                                      Hits      TTFB Avg  TTLB Avg
=========================================================================
GET /pfc2                                 17        555.29    804.35
GET /pfc2/                                17        162.06    162.06
GET /pfc2/css/telematica.css              17        150.65    150.76
GET /Icons/valid-xhtml11-blue             17        133.00    133.00
GET /pfc2/imagenes/telematica/            17        139.29    139.29
GET /pfc2/imagenes/telematica/            16        148.44    148.44
GET /pfc2/ServletPrincipal                16        162.88    163.69
POST /pfc2/AutenServlet                   16        141.81    142.94
GET /pfc2/VerReuniones                    16        220.31    1805.69
```

Figura 6.8: Resumen de los tiempos medios de respuesta en el escenario 2 con 5 clientes

Escenario 2: 10 clientes

```
Socket Statistics
-----------------------------------------
Socket Connects:              190
Total Bytes Sent (in KB):     116.69
Bytes Sent Rate (in KB/s):    1.94
Total Bytes Recv (in KB):     1157.18
Bytes Recv Rate (in KB/s):    19.29
```

Figura 6.9: Estadísticas de los parámetros de la conexión en el escenario 2 con 10 clientes

```
Page Summary
Page                                      Hits      TTFB Avg  TTLB Avg
=========================================================================
GET /pfc2                                 19        415.00    623.32
GET /pfc2/                                19        164.47    164.47
GET /pfc2/css/telematica.css              19        153.11    153.11
GET /Icons/valid-xhtml11-blue             19        143.95    143.95
GET /pfc2/imagenes/telematica/            19        149.84    149.84
GET /pfc2/imagenes/telematica/            19        147.79    147.84
GET /pfc2/ServletPrincipal                19        152.79    152.89
POST /pfc2/AutenServlet                   19        154.32    155.53
GET /pfc2/VerReuniones                    19        223.11    1451.42
```

Figura 6.10: Resumen de los tiempos medios de respuesta en el escenario 2 con 10 clientes

Escenario 2: 20 clientes

```
Socket Statistics
-----------------------------------------
Socket Connects:              163
Total Bytes Sent (in KB):     100.04
Bytes Sent Rate (in KB/s):    1.67
Total Bytes Recv (in KB):     1109.64
Bytes Recv Rate (in KB/s):    18.49
```

Figura 6.11: Estadísticas de los parámetros de la conexión en el escenario 2 con 20 clientes

```
Page Summary
Page                                        Hits      TTFB Avg  TTLB Avg
=============================================================================
GET /pfc2                                   17        565.12    850.94
GET /pfc2/                                  17        198.94    198.94
GET /pfc2/css/telematica.css                16        168.75    168.75
GET /Icons/valid-xhtml11-blue               16        151.31    151.31
GET /pfc2/imagenes/telematica/              16        146.00    146.00
GET /pfc2/imagenes/telematica/              16        141.38    141.38
GET /pfc2/ServletPrincipal                  16        145.25    145.50
POST /pfc2/AutenServlet                     16        154.25    155.25
GET /pfc2/VerReuniones                      16        249.63    1712.13
```

Figura 6.12: Resumen de los tiempos medios de respuesta en el escenario 2 con 20 clientes

6.1.3.3 Tercer escenario: Creación de una reunión, notificación a los invitados y almacenamiento de los detalles en la base de datos.

Escenario 3: 5 clientes

```
Socket Statistics
-----------------------------------------
Socket Connects:              121
Total Bytes Sent (in KB):     74.56
Bytes Sent Rate (in KB/s):    1.24
Total Bytes Recv (in KB):     656.21
Bytes Recv Rate (in KB/s):    10.94
```

Figura 6.13: Estadísticas de los parámetros de la conexión en el escenario 3 con 5 clientes

Page Summary

Page	Hits	TTFB Avg	TTLB Avg
GET /pfc2	6	124.00	282.17
GET /pfc2/	6	134.33	134.33
GET /pfc2/css/telematica.css	6	149.17	149.17
GET /Icons/valid-xhtml11-blue	6	125.17	125.17
GET /pfc2/imagenes/telematica/	6	142.83	142.83
GET /pfc2/imagenes/telematica/	6	139.33	139.33
GET /pfc2/ServletPrincipal	6	150.83	151.67
POST /pfc2/AutenServlet	6	161.00	162.00
GET /pfc2/NuevaReun	6	137.33	138.33
GET /pfc2/UnaFecha	6	157.50	422.17
GET /pfc2/css/calendar-blue.cs	6	199.83	310.50
GET /pfc2/jscript/calendar/cal	6	155.17	1356.50
GET /pfc2/jscript/calendar/lan	6	461.17	480.00
GET /pfc2/jscript/calendar/cal	6	258.83	521.67
GET /pfc2/imagenes/bombilla.gi	6	190.83	273.00
GET /pfc2/css/menuarrow.gif	6	142.50	142.50
POST /pfc2/EventDia	6	155.83	251.17
POST /pfc2/DatosEvent	6	160.17	234.83
POST /pfc2/GuardarEvento	6	4560.83	4560.83

Figura 6.14: Resumen de los tiempos medios de respuesta en el escenario 3 con 5 clientes

Escenario 3: 10 clientes

```
Socket Statistics
-----------------------------------------
Socket Connects:              113
Total Bytes Sent (in KB):     69.39
Bytes Sent Rate (in KB/s):    1.16
Total Bytes Recv (in KB):     590.70
Bytes Recv Rate (in KB/s):    9.85
```

Figura 6.15: Estadísticas de los parámetros de la conexión en el escenario 3 con 10 clientes

```
Page Summary
Page                                  Hits      TTFB Avg  TTLB Avg
=============================================================================
GET /pfc2                              6         190.83    404.50
GET /pfc2/                             6         221.50    221.50
GET /pfc2/css/telematica.css           6         212.00    212.00
GET /Icons/valid-xhtml11-blue          6         164.83    164.83
GET /pfc2/imagenes/telematica/         6         162.33    162.33
GET /pfc2/imagenes/telematica/         6         163.67    163.67
GET /pfc2/ServletPrincipal             6         185.00    186.17
POST /pfc2/AutenServlet                6         163.83    164.83
GET /pfc2/NuevaReun                    6         162.67    163.67
GET /pfc2/UnaFecha                     6         196.00    491.83
GET /pfc2/css/calendar-blue.cs         6         249.50    371.50
GET /pfc2/jscript/calendar/cal         5         229.60    1511.00
GET /pfc2/jscript/calendar/lan         5         466.60    485.20
GET /pfc2/jscript/calendar/cal         5         302.00    612.00
GET /pfc2/imagenes/bombilla.gi         5         262.60    382.60
GET /pfc2/css/menuarrow.gif            5         248.20    248.20
POST /pfc2/EventDia                    5         241.60    359.40
POST /pfc2/DatosEvent                  5         209.00    306.40
POST /pfc2/GuardarEvento               5         4735.00   4735.00
```

Figura 6.16: Resumen de los tiempos medios de respuesta en el escenario 3 con 10 clientes

Escenario 3: 20 clientes

```
Socket Statistics
-----------------------------------------
Socket Connects:              111
Total Bytes Sent (in KB):     68.45
Bytes Sent Rate (in KB/s):    1.14
Total Bytes Recv (in KB):     565.63
Bytes Recv Rate (in KB/s):    9.43
```

Figura 6.17: Estadísticas de los parámetros de la conexión en el escenario 3 con 20 clientes

```
Page Summary
Page                                        Hits      TTFB Avg  TTLB Avg
=============================================================================
GET /pfc2                                   6         197.33    381.50
GET /pfc2/                                  6         155.33    155.33
GET /pfc2/css/telematica.css                6         164.33    164.33
GET /Icons/valid-xhtml11-blue               6         143.67    143.67
GET /pfc2/imagenes/telematica/              6         181.50    181.50
GET /pfc2/imagenes/telematica/              6         185.50    185.50
GET /pfc2/ServletPrincipal                  6         215.83    216.83
POST /pfc2/AutenServlet                     6         182.33    183.50
GET /pfc2/NuevaReun                         6         178.17    179.17
GET /pfc2/UnaFecha                          5         213.80    520.80
GET /pfc2/css/calendar-blue.cs              5         239.20    368.00
GET /pfc2/jscript/calendar/cal              5         196.40    1516.00
GET /pfc2/jscript/calendar/lan              5         531.20    549.60
GET /pfc2/jscript/calendar/cal              5         299.80    576.60
GET /pfc2/imagenes/bombilla.gi              5         234.60    355.20
GET /pfc2/css/menuarrow.gif                 5         213.40    213.40
POST /pfc2/EventDia                         5         232.80    348.20
POST /pfc2/DatosEvent                       5         244.60    356.80
POST /pfc2/GuardarEvento                    5         4981.40   4981.40
```

Figura 6.18: Resumen de los tiempos medios de respuesta en el escenario 3 con 20 clientes

Como conclusiones a estas pruebas y como era de esperar, se puede obtener que según va aumentando el número de clientes concurrentes en la aplicación la respuesta por parte del servidor a cada una de estas peticiones se hace más lenta. No obstante, a la vista de los resultados obtenidos se puede entrever que la caída de rendimiento no parece ser significativa. Esto último se puede apreciar en el escenario más costoso donde se ve que al petición POST /pfc2/GuardarEvento está entre los 4.5 sg y 4.9 sg para los 5, 10 y 20 clientes.

Además, tal y como se dijo al principio, se puede corroborar que las operaciones que menos recursos consumen son las que simplemente obtienen datos almacenados y los muestran en el navegador, seguidas de las que realizan modificaciones sobre la base de datos y por ultimo las que envían un correo electrónico, ya que es necesario establecer una conexión con un servidor externo y enviar los mensajes a través de él.

6.2 Conclusiones

El objetivo principal de este proyecto era el desarrollo de una herramienta que ayudara a la organización de reuniones contemplando las disponibilidades de las personas que van a asistir a esa reunión.

Así mismo, se ha conseguido que esta herramienta disponga de un mecanismo de notificaciones que permite avisar a una persona si ha sido invitada además de darle acceso para tomar la decisión sobre el día y la hora en la que se celebra la reunión e igualmente avisarla de cualquier cambio.

Una vez terminado este proyecto es necesario echar un vistazo a los objetivos planteados al principio de este para observar hasta que punto se han cumplido las expectativas:

Primero se va a proceder a revisar los objetivos generales de la aplicación para después ver en detalle los objetivos más específicos de Rendez-Vous.

1. *Acceso vía Web con cualquier navegador y con cualquier Sistema Operativo.*
2. *GUI sencilla.*
3. *Cumplimiento de las normas establecidas conforme a XHTML 1.1 y nivel de accesibilidad de la herramienta AA.*

Gracias a los estándares de XHTML, lo que se consigue es que cualquier usuario pueda acceder a Rendez-Vous sin que importe el tipo de navegador que está usando. Además, los diferentes contenidos de las páginas Web de la aplicación han sido modificados para satisfacer el nivel de verificación de accesibilidad ("AA"). De no haber sido así, los usuarios con alguna discapacidad podrían encontrar alguna dificultad para acceder a la información. Aplicando este nivel de verificación se mejora notablemente la accesibilidad del sitio Web a los usuarios con discapacidades y beneficia igualmente al resto de los usuarios.

Además del uso en navegadores convencionales para PC, se probó también que la aplicación funcionase en tres de los dispositivos que actualmente engloban a los más usados en el mercado de terminales móviles: Blackberry, Nokia N95 e IPod Touch/IPhone 2G.

El resultado fue que Rendez-Vous funcionaba en los tres pero con un resultado visual de cara al usuario distinto:

- En una Blackberry se vio que gracias al XHTML 1.1 la aplicación funcionaba bien, pero con alguna carencia visual debida que no se ejecuta ni las hojas de estilo ni el Javascript.
- En el Terminal Nokia 95 se observó que tanto la aplicación como los CSS y Javascripts se ejecutan perfectamente, pero que debido al propio navegador del teléfono, el uso sobre éste es algo incomodo ya que no se llega a mostrar la pagina Web en su totalidad, sino sólo una parte de ella.
- Finalmente, en el IPod Touch el resultado obtenido era igual al que se obtiene en un navegador de los que se pueden usar en cualquier ordenador. Todo ello es debido a que implementa un sistema de navegación más potente basado en el motor Safari y a una gestión del acceso a Internet más orientada al sistema de los navegadores

tradicionales que al de los dispositivos móviles. No se aprecia carencia visual alguna en lo referente a la ejecución sin errores de Javascript y de las hojas de estilo.

Rendez-Vous no se pensó inicialmente para su uso sobre este tipo de dispositivos, pero gracias al cumplimiento de los estándares del XHTML1.1+CSS se ha podido comprobar estos resultados.

Así mismo y gracias a la portabilidad de Java, Rendez-Vous puede correr sobre cualquier tipo de sistema operativo y aunque ahora mismo esta ejecutándose sobre una distribución Linux basada en Ubuntu 8.04, la aplicación esta probada sobre entornos Windows y Macintosh conservando todas sus funcionalidades y prestaciones.

4. *Realizar el desarrollo basándose en componentes de código libre.* El uso de componentes basados en código libre no implica que las tecnologías usadas no sean eficaces. La elección de usar tecnologías como Java, MySQL, y Tomcat se debe a que gracias a su política de código abierto, se ha podido consultar libremente sus APIs de referencia en cualquier momento que se haya necesitado, teniendo el soporte de una amplia comunidad de usuarios. Además no ha sido necesario preocuparse de ninguna licencia ya fuese a la hora del desarrollo de la aplicación o a la hora de la implementación.

Seguidamente, se procederá a analizar los objetivos para la aplicación Rendez-Vous:

1. *Facilidad a la hora de organizar e invitar gente a una reunión así como a la hora de expresar las disponibilidades para dicha reunión.* Estos son los objetivos principales del proyecto. En base a ellos se ha diseñado e implementado *Rendez-Vous* intentando siempre que a un usuario no se le presenten problemas a la hora de usar la aplicación, tanto para organizar como de votar las fechas de una reunión.

2. *Poder proporcionar tantas opciones como se desee a la hora de proponer fechas para una reunión.* Una de las limitaciones que se encontraron en la aplicación *Meetingwizard* es que en su distribución gratuita, la posibilidad que se ofrecía para la creación de las reuniones no se podía proponer más de 10 fechas por reunión. En *Rendez-Vous* se ha intentado solventar esto para que los usuarios puedan proponer tantas fechas como estimen oportunas.

3. *Posibilidad de contemplar una respuesta por parte de un invitado que no sea ni afirmativa ni negativa.* Tanto en *MeetingWizard* como en *Doodle*, a la hora de que los invitados expresen sus preferencias y disponibilidades estos se encuentran con que tan solo se contemplan respuestas afirmativas y negativas. Aquí se ha querido abrir un poco más el abanico de opciones de respuesta añadiendo una opción adicional neutra ya que una persona no siempre sabe con toda seguridad si estará disponible. De esta forma se puede notificar al menos que ha recibido la invitación y lo tiene en mente pero sin decidir.

4. *Proporcionar distintas interfaces Web mediante el uso de CSS para demostrar la flexibilidad del producto, algunas de índole corporativa, pero con posibilidad de usar otras más estéticas.* A través de las opciones que el propio navegador proporciona se puede escoger la vista que se desee tener de la aplicación. Con *Rendez-Vous* se proporciona una vista diseñada especialmente para el Departamento de Telemática de la Universidad Carlos III y una vista modelo más informal en la que se pueden basar futuras nuevas vistas.

5. *Posibilidad de envío de notificaciones ya sea para avisar de la creación de una nueva reunión o para recordarla mediante un servidor SMTP y correos electrónicos.* Este objetivo se cumple usando la clase `Email.java` que a su vez se apoya en *Javamail*. Gracias a todo ello la aplicación se conecta a un servidor SMTP especificado y es capaz de enviar correos electrónicos. Las clases encargadas de generar los mensajes apropiados para cada situación se pueden consultar en la sección 5.1.

6. *Posibilidad de que las claves sean encriptadas para dar mayor seguridad a la hora de almacenar las contraseñas de los usuarios o las URL de referencia.* Tanto la contraseña de acceso como la clave incluida en las invitaciones de los usuarios a las reuniones están codificadas mediante SHA-1 que es uno de los algoritmos más seguros actualmente de codificación.

7. *Permitir a los invitados poner sus preferencias estén o no estén dados de alta en el sistema.* Un invitado puede no estar dado de alta cuando se le invita a una reunión, por lo que para evitar este problema lo que se hace es que la referencia a ese invitado se lleva a cabo mediante su correo electrónico. Además, gracias a la clave incluida en la invitación que se recibe por correo electrónico, cada invitado puede acceder al sistema de forma unívoca y reflejar si asistirá o no a una reunión y/o expresar las fechas que le vienen mejor sin que para ello tenga que estar registrado en la aplicación.

8. *Posibilidad de exportar las reuniones pendientes a un fichero con extensión .ics, ya sean aquellas reuniones creadas por el propio usuario como a las que ha sido invitado. De este modo aplicaciones como ICAL de Mac o Outlook de Microsoft serán capaces de importarlas en sus respectivos calendarios. Así mismo, importar ficheros .ics que generen estos programas para crear las reuniones.* Después de estudiar el RFC 2445 que recoge el estándar del ICal, se adaptó este proyecto para que pudiese exportar las reuniones que tiene un usuario, a un fichero .ics. El resultado es que dicho usuario obtiene el fichero **rendezvous.ics** donde se obtendrá una lista con aquellas reuniones que el usuario haya creado y aquellas a las que habiendo sido invitado, haya confirmado su asistencia. Además, para que una reunión se añada al listado generado, su fecha de celebración debe haber sido confirmada por su organizador correspondiente.

Por otra parte, y debido a que las especificaciones de dicho estándar no contemplan la posibilidad de guardar en el calendario citas que no estén confirmadas (todas las citas deben de tener una única fecha de celebración) se decidió no implementar esta funcionalidad.

Por último, se procederá a comparar Rendez-Vous con otras herramientas ya existentes en el mercado como son MeetingWizard *[1]* y Doodle *[2],* de las cuales se sacó la idea de este proyecto y Google Calendar *[22]* que salió al mercado mientras se estaba desarrollando esta idea. En esta comparación se han tomado los puntos más relevantes de ellas y en los que sobresalen unas aplicaciones frente a otras:

	MeetingWizard	Doodle	Google Calendar	Rendez-Vous
Posibilidad de crear invitaciones sin registrarse	NO	SI	NO	NO
Posibilidad de evento confirmado o múltiples fechas	SI	SI	NO	SI
Facilidad para introducir fechas y horas	SI	NO	SI	SI
Posibilidad de distintas duraciones dependiendo de la fecha	NO	NO	NO	SI
Notificación por correo electrónico	SI	NO	SI	SI
Posibilidad de introducir los contactos de una agenda	SI	NO	SI	NO
Varias vistas de la aplicación	NO	NO	NO	SI
Posibilidad de consultar las reuniones pendientes	SI	NO	SI	SI
Sincronización de calendarios externos mediante ICAL	NO	NO	SI	SI
Uso gratuito	NO	SI	SI	SI

Figura 6.19: Tabla comparativa entre las 4 aplicaciones

Teniendo como referencia la figura 6.19, la primera característica que casi todas las herramientas tienen, exceptuando a Doodle, es el requisito de que el administrador de la reunión debe de estar dado de alta y autenticado. Doodle da la posibilidad de que dicho administrador esté registrado o no a la hora de crear una nueva reunión, pero la ventaja que ofrece al estar autenticado es la posibilidad de gestionar las reuniones pendientes tal y como hacen el resto de herramientas.

Como segundo punto, se valora uno de los requisitos claves de este proyecto: la posibilidad de generar una reunión con una fecha que no esté confirmada y decidirla después de recibir todas o la mayoría de respuestas por parte de los invitados. De este modo se consigue que encaje en el mayor número posible de agendas. Solamente Google Calendar se queda fuera de esta característica debido a que se apoya en la política de que primero es un calendario en el que se apuntan eventos ya confirmados, y después se ofrece la posibilidad de invitar a gente a esos eventos como les pasa a otras herramientas como ICal o Outlook.

El tercer punto contempla que el diseño de la aplicación ayude al usuario a introducir cualquier fecha y hora de manera sencilla cuando sea necesario. A la hora de seleccionar la fecha, las cuatro herramientas proporcionan un calendario que permite cómodamente elegir una fecha, mientras que cuando hay que seleccionar las horas, Doodle se queda un poco pobre debido a que solo ofrece la posibilidad de pedir al usuario que siga un patrón establecido introduciéndolas en un cuadro a mano.

En el cuarto punto lo que se contempla es la posibilidad de introducir distintas duraciones en la reunión dependiendo de la fecha que se plantee para ello. Ninguna de las herramientas analizadas salvo Rendez-Vous contempla esta opción.

El quinto punto habla sobre la posibilidad de que la propia aplicación sea quien envíe un correo electrónico a cada uno de los invitados. Tres de las cuatro herramientas incorporan esta funcionalidad, pero Doodle vuelve a quedarse un poco austera en este sentido debido a que simplemente proporciona un enlace que el administrador de la reunión debe de ocuparse de remitir a cada uno de los invitados que considere pertinentes.

Sobre la posibilidad de importar los contactos de una agenda a la hora de decidir los invitados que se menciona en el punto seis, hay que decir que éste es el único punto en el que Rendez-Vous ha quedado por debajo de las otras herramientas ya que los invitados deben de ser introducidos manualmente.

Por otra parte, y tal y como se menciona en el séptimo punto, Rendez-Vous es la única herramienta de las aquí comparadas que incorpora la posibilidad de que el usuario decida el tipo de vista que quiere utilizar, si una de carácter corporativo o otra de carácter más informal usando para ello el propio navegador.

El octavo punto contempla la posibilidad de que el administrador pueda ver a través de la aplicación las reuniones que tienen pendientes. Como se puede observar, tanto MeetingWizard como Google Calentar y Rendez-Vous ofrecen esta posibilidad pero en cambio para que Doodle la ofrezca, el usuario debe de estar registrado tal y como se comentó en el primer punto.

El noveno punto se debe a la posibilidad de sincronizar las reuniones que un usuario administrador tiene pendientes mediante un fichero .ics con calendarios externos como pueden ser ICal de Mac, el calendario de Outlook o el propio Google Calendar. Como es lógico, Google Calendar incorpora esta opción para su propia sincronización. Rendez-Vous la incorpora también cumpliendo los estándares de ICalendar y haciendo compatible este fichero con los calendarios anteriormente mencionados.

Finalmente, se valora el uso gratuito de la aplicación. Las tres herramientas aquí consideradas que están en el mercado ofrecen un uso gratuito, aunque para usar todas las funcionalidades de MeetingWizard es necesario adquirir una licencia ampliando el uso de las operaciones existentes. Por parte de Rendez-Vous se pretende que este uso sea también gratuito aunque esté dirigido al Departamento de Telemática y sean ellos quienes lo administren y gestionen.

Una pregunta que puede asaltar viendo todo esto es: *¿Por qué alguien debería de usar Rendez-Vous existiendo otras aplicaciones del mismo formato en el mercado?* La razón principal es que Rendez-Vous ha intentado reunir todas las ideas buenas que este tipo de aplicación posee y juntarlas en una sola herramienta para poder ofrecérselas a un usuario.

Por otra parte, y como ya se comentó, Rendez-Vous está basada en componentes de código libre, con lo que se pretende ofrecer además la posibilidad de dejar el entorno abierto para que un futuro se pueda incorporar mejoras sobre la aplicación así como nuevas funcionalidades que se vayan necesitando con el tiempo.

6.3 Trabajos futuros

Echando la vista atrás se ha visto como los principales objetivos planteados para Rendez-Vous se han cumplido. No obstante, a lo largo del desarrollo de este proyecto se han abierto nuevas vías de investigación que ayudarían a que esta aplicación fuera más completa:

1. Rendez-Vous permite proponer varias fechas para organizar una reunión, pero podría darse el caso en el que el problema no sea la fecha de celebración del mismo, sino el lugar donde se lleve a cabo. Rendez-Vous podría ser mas completo si incorporase esta opción.

2. Por otra parte, también puede llegar a darse el caso en el que se invite a un usuario a una reunión, reciba la invitación y se encuentre con que no tiene disponibilidad en ninguna de las fechas propuestas. Actualmente se cuenta con un campo comentario donde los invitados pueden expresar esto y otras cuestiones, pero el programa podría ser mas completo si los mismos usuarios invitados a un evento pudiesen proponer nuevas fechas que encajen dentro de sus planes.

3. Gracias a Internet es posible mantener un encuentro con una persona que este en la otra punta del mundo. Actualmente, las horas establecidas en Rendez-Vous se tienen en cuenta como hora local de Madrid (*Central Europe Time*), o en su defecto la zona horaria propia del servidor donde físicamente se halle instalada la aplicación. Para que un usuario que está en un lugar remoto no tenga que preocuparse de realizar la conversión a su hora local del momento en el que se celebre el evento, se podría incorporar un campo que calculase la diferencia entre esos usos horarios dándole al usuario de ese modo la información de la reunión en su hora local.

4. También se podría hacer que la aplicación fuese accesible por gente que no sea hispanohablante. Para ello se podría incorporar una opción que permita elegir el idioma en que se muestra la aplicación.

5. Del mismo modo se podría ofrecer a los usuarios una agenda en la que almacenar sus contactos favoritos y desde la cual pudiesen seleccionar a los invitados de una reunión de una forma más ágil. Además sería interesante la opción de importar/exportar sus contactos desde/a un fichero externo.

Si se decidiese que Rendez-Vous ya cumple las expectativas necesarias se podrían abrir también nuevas vías de investigación con aplicaciones externas a ella:

1. Rendez-Vous es una herramienta que ayuda a un grupo de personas a ponerse de acuerdo sobre la fecha en la que celebrar un evento. Un módulo adicional muy útil sería poder disponer una agenda electrónica que se pudiera consultar vía Web y que se sincronizase con Rendez-Vous, encargándose dicho módulo de sincronizar automáticamente las reuniones con fechas ya confirmadas a las que alguien es invitado. Además, gracias a este nuevo modulo se podrían importar ficheros creados con aplicaciones externas que cumplan el formato ICal.

2. Por otra parte, no a todo el mundo le gusta acceder vía navegador. Se podría desarrollar un interfaz que se instalase en el ordenador del usuario y se conectase con el servidor para poder trabajar en el equipo de forma local. Así mismo y debido a la creciente demanda de acceso móvil a Internet sería bueno que el acceso a Rendez-Vous se optimizase para su uso en dispositivos móviles como teléfonos, PDAs, etc. Se ha podido observar que en la situación actual, la visualización es aceptable y operativa pero sujeta a los requerimientos propios impuestos por los diferentes navegadores. Lo ideal sería hacer desarrollos específicos en forma de aplicaciones para cada una de las tecnologías en las que se implementan los diversos terminales móviles. Esto mejoraría la manejabilidad, la visualización y la interacción con el usuario.

Apéndice A

Manual de Instalación

En este apéndice se va a proceder a describir los pasos necesarios para poder poner en marcha a *Rendez-Vous*. Primero se hará una breve referencia sobre cómo instalar las herramientas para su desarrollo y a continuación se procederá a explicar como instalar todos los componentes necesarios para que la aplicación en sí, funcione sobre un sistema Ubuntu Linux.

A.1 Herramientas de desarrollo

Para poder llevar a cabo el desarrollo de la aplicación se ha utilizado el editor Netbeans. Es un editor muy completo que se apoya en Java y Tomcat, gracias a los cuales se ha compilado la aplicación y se ha desplegado en el servidor.

A.2 Instalación de las aplicaciones necesarias en el servidor

Pasamos a explicar cómo instalar la aplicación sobre un sistema Ubuntu Linux. Se entiende que se cuenta con el conocimiento suficiente para instalar el sistema Ubuntu y la aplicación Java.

Primero se accede a la página oficial de “MySQL” y en el apartado “Community” se procede a descargar el “MySQL Standard Server” que mas se ajuste a nuestro entorno. La versión Standard debería cubrir con creces las necesidades para la aplicación. Para facilitar la administración de la base de datos se recomienda también descargarse las “GUI Tools”. Una vez descargado e instalado, es necesario crear una base de datos llamada ***“rendezvous”*** dentro de MySQL***.*** Se recomienda crear un usuario distinto al usuario “root” pero que tenga todos los permisos sobre esa base de datos.

Para configurar el servidor “Apache Tomcat” en un entorno linux, es necesario como paso previo, bajarse el archivo “*apache-tomcat-xxx.tar.gz”* de la página oficial. Tras ello lo descomprimimos y seguimos la siguiente receta:

- Establecer la variable de entorno CATALINA_HOME para que apunte al directorio donde se ha instalado el servidor Tomcat.
- Establecer la variable de entorno JAVA_HOME para que apunte al directorio donde está instalado Java.
- Añadir los directorios
  ${JAVA_HOME}/bin
  ${CATALINA_HOME}/bin
  a la variable de entorno PATH del sistema operativo.

Seguidamente debemos tener presente que para arrancar y parar el servidor, es necesario ejecutar los scripts:

startup.sh
shutdown.sh

desde una shell. Dichos scripts se encuentran en el directorio bin de la instalación del Tomcat.

Para desplegar un Servlet es necesario crear un contexto de la aplicación Web. Esto se consigue creando un directorio debajo de:

${CATALINA_HOME}/webapps/

Este directorio se corresponde con el directorio raíz de nuestra aplicación web y por lo tanto debajo de él, debemos crear la estructura de subdirectorios indicada anteriormente.

Hay que tener muy presente que el nombre del contexto, es el primer nivel de la jerarquía de la ruta de acceso a nuestra aplicación y que en este caso será:

http://localhost:8080/prueba/path_recurso

```
${CATALINA_HOME}/webapps +-- prueba+-- WEB-INF+-- web.xml
                                      |        +--classes +-HolaMundo.class
                                      +-- estatico.html
                                      |
                                      +-- tomcat.gif
```

A.3 Adaptación del código de Rendez-Vous para un nuevo entorno

A.3.1: Creación de las tablas en la base de datos.

Antes de nada, es necesario volcar el fichero *RedezVousBD.sql* que se proporciona en el CD de instalación sobre la base de datos ***rendezvous*** creada a partir de las especificaciones dadas anteriormente. Para ello se puede hacer a través de la opción “abrir script” que se encuentra en **MySQL Query Browser**. No obstante, si no se ha instalado la GUI de MySQL, también se puede volcar el fichero mediante el comando:

```
mysql -u usuario -p -D rendezvous < \RendezVousBD.sql
```

Se nos pedirá la clave del usuario proporcionado y se procederá a crear todas las tablas de la base de datos. Es necesario que el usuario posea los permisos necesarios para poder crear dichas tablas en la base de datos **rendezvous.**

A.3.2: Adaptación del código al nuevo entorno

Para poder disfrutar de Rendez-Vous en un entorno personalizado y con una configuración distinta a la proporcionada por defecto es necesario adaptar algunos parámetros de los ficheros del código:

A.3.2.1 Conexion.java

Como ya se explicó anteriormente, la clase `Conexion.java` se encarga de establecer la comunicación con la base de datos que esté instalada en una determinada máquina. Para ello es necesario especificar el nombre de usuario junto con su contraseña para acceder a la base de datos. Del mismo modo se debe proporcionar la url y el puerto donde es accesible dicha base de datos:

```
protected Connection con;
private String driver = "com.mysql.jdbc.Driver";
private String user = "marta";
private String password = "marta";
private String url="jdbc:mysql://localhost:3306/rendezvous";
```

A.3.2.2 Email.java

`Email.java` se conecta con un servidor SMTP para poder enviar los correos electrónicos necesarios para cada notificación. Es necesario configurar este fichero para que estas notificaciones funcionen correctamente.

Primero hay que definir la dirección de correo electrónico desde la que se enviarán los mensajes. Seguidamente es necesario poner la dirección IP que identifique al servidor SMTP. Con objeto de facilitar la referencia en las URLs que se manden se debe definir donde está la corriendo la aplicación. Es necesario configurar correctamente este paso para que las URL que luego se manden en los mensajes aparezcan correctamente:

```
public Email() {
        from = "rendezvous@dominio.com";
        host = "163.41.227.156";//Servidor SMTP
        //Servidor donde corre el Tomcat
        hostTomcat = "http://rendezvous.com:8088";
     }
```

Por otra parte, si el servidor SMTP no es autenticado dejamos comentariada la línea:

```
prop.put("mail.smtp.auth", "true");
```

localizada en el método `enviar()`.

Si por el contrario hace falta autenticarse contra el servidor, vamos a necesitar configurar la clase SMTPAutentication con los parámetros de usuario y contraseña de una cuenta válida en el servidor SMTP, por lo que en su constructor haremos:

```
public PasswordAuthentication getPasswordAuthentication() {

     String username = "marta@dominio.com";
     String password = "password";

     return new PasswordAuthentication(username, password);

    }
```

indicándole con las variables username y password los datos de autenticación para hacer SMTP Relay.

Apéndice B

Introducción a la Aplicación

Rendez-Vous es una aplicación diseñada e implementada para ayudar a un grupo de gente a ponerse de acuerdo a la hora que celebrar una reunión, intentando facilitar dicha labor y evita toda clase de problemas que se suelen derivar.

Primero, se crea una reunión con unos invitados y fechas para la celebración de esta.

Cada uno de los invitados recibe una invitación a la reunión creada en a través de un correo electrónico. Así mismo, en el mismo mensaje recibe una URL a donde el invitado puede dirigirse para expresar las preferencias en cada una de las fechas propuestas. Ya que un invitado es referenciado a través de su correo electrónico, no es necesario que éste esté dado de alta en la aplicación para exponer sus preferencias acerca de una fecha.

Finalmente, el organizador del evento puede decidir cuando celebrar la reunión viendo las respuestas de cada uno de los invitados.

Antes de nada, es necesario acceder mediante un usuario y una contraseña a la aplicación.

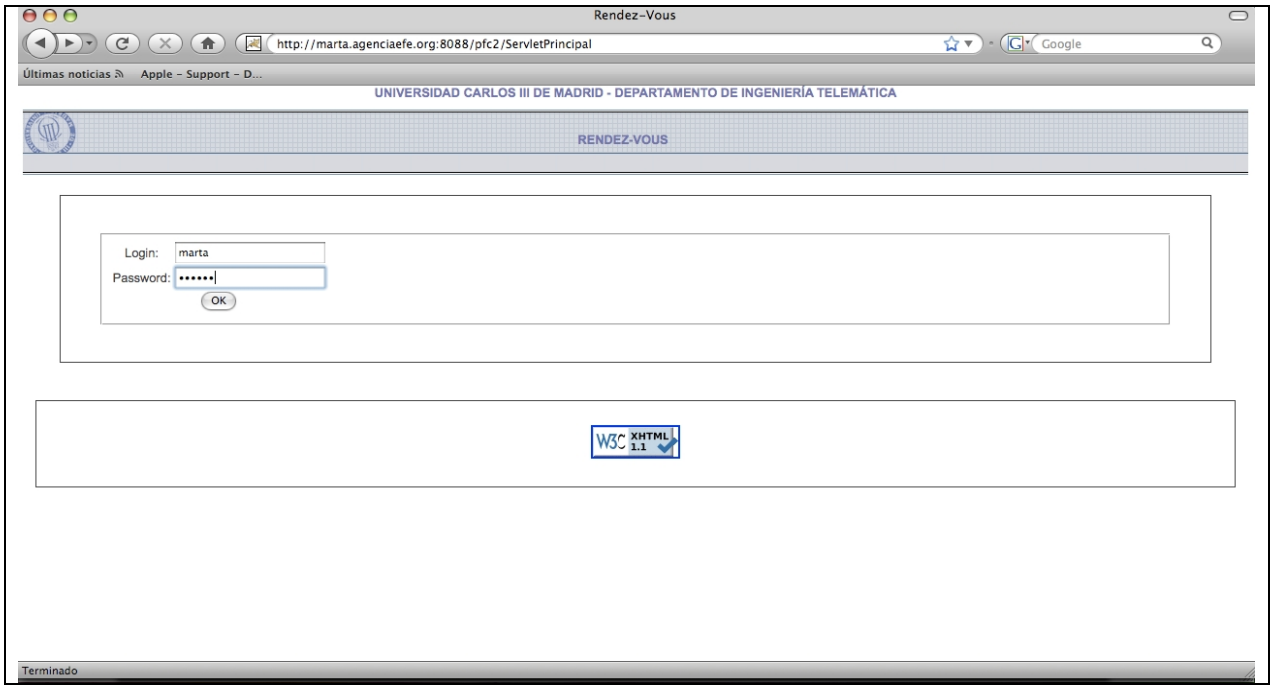

Figura B.1: Ventana de acceso a Rendez-Vous

Una vez dentro de la aplicación, el usuario dispone de tres opciones en el menú principal:

- Crear una nueva reunión.
- Consultar las Reuniones que ha creado o a las que ha sido invitado.
- Cambiar la información personal.

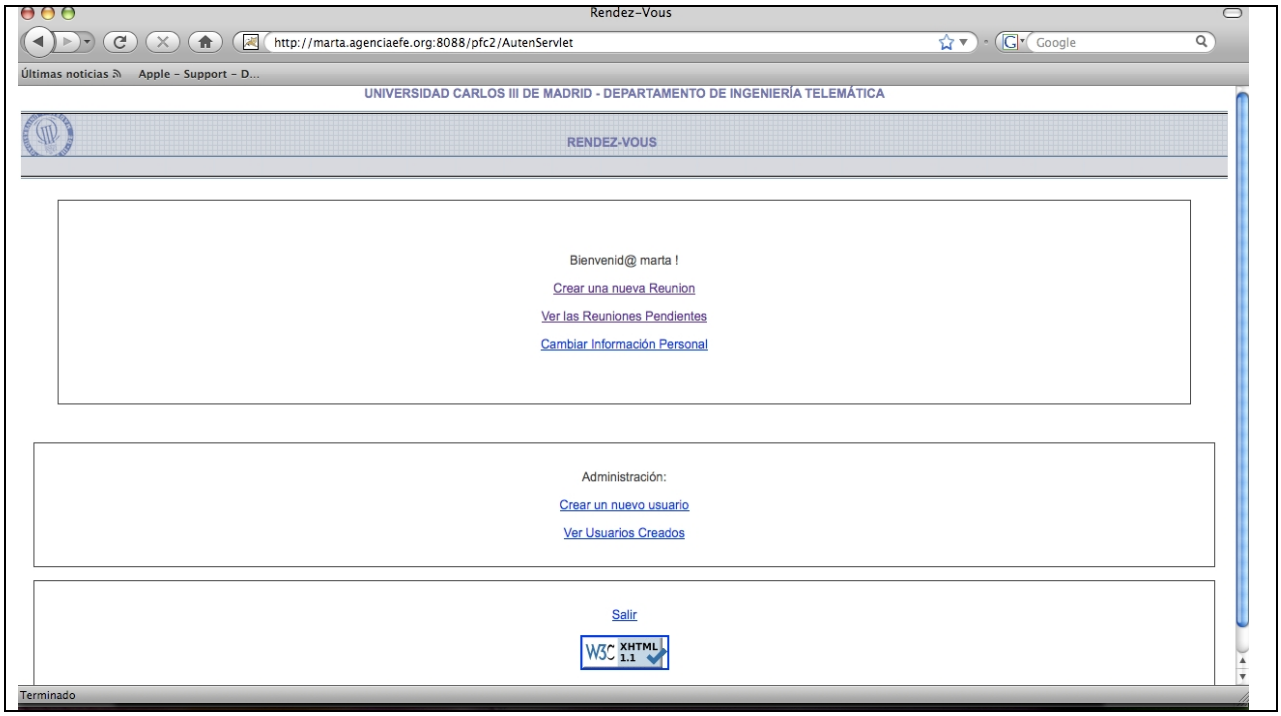

Figura B.2: Menú Principal de Rendez-Vous

B.1 Creación de una reunión

A través de esta opción, el usuario puede crear una reunión con una fecha confirmada o con varias opciones de fecha, para que dependiendo de la disponibilidad de sus invitados pueda elegir la fecha que mejor encaje entre todos.

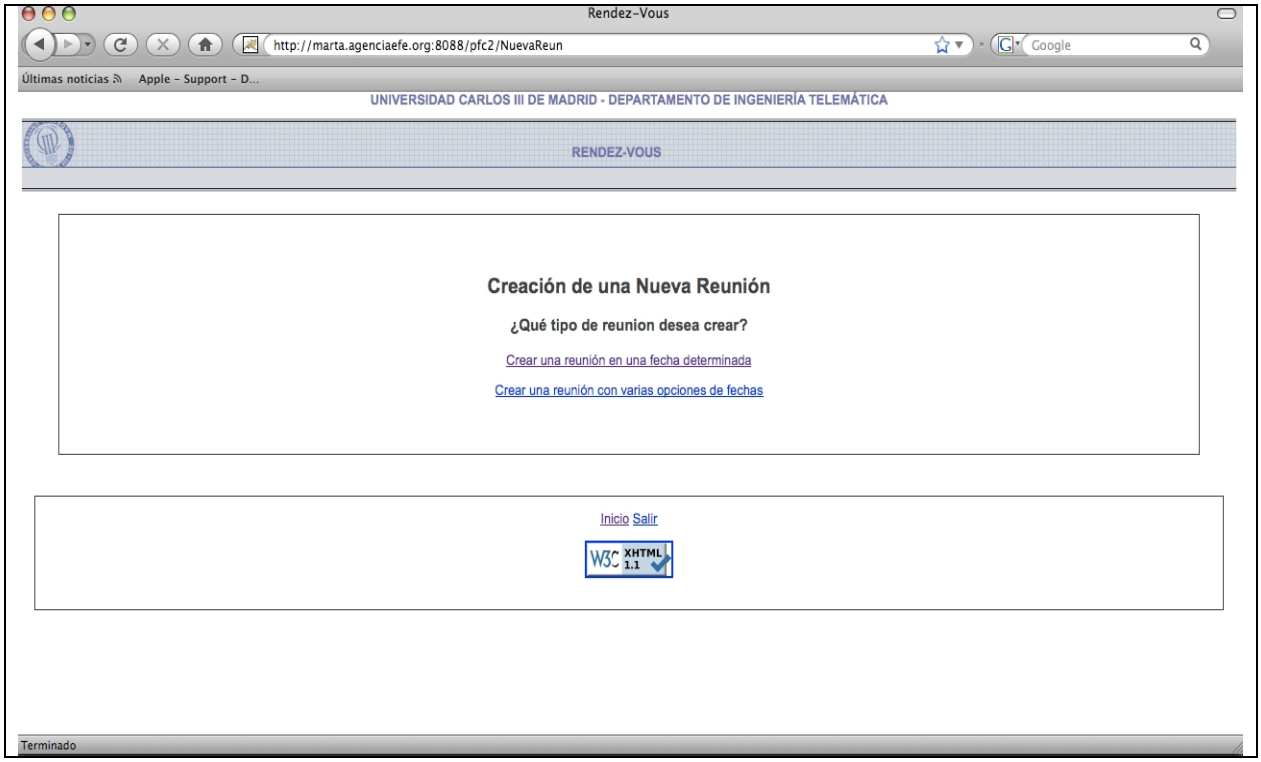

Figura B.3: Menú para la creación de una reunión.

Con calendario de Rendez-Vous se puede definir la fecha o fechas en las que se celebrará la reunión dependiendo de si éstas son confirmadas o no en la que la reunión se celebrará.

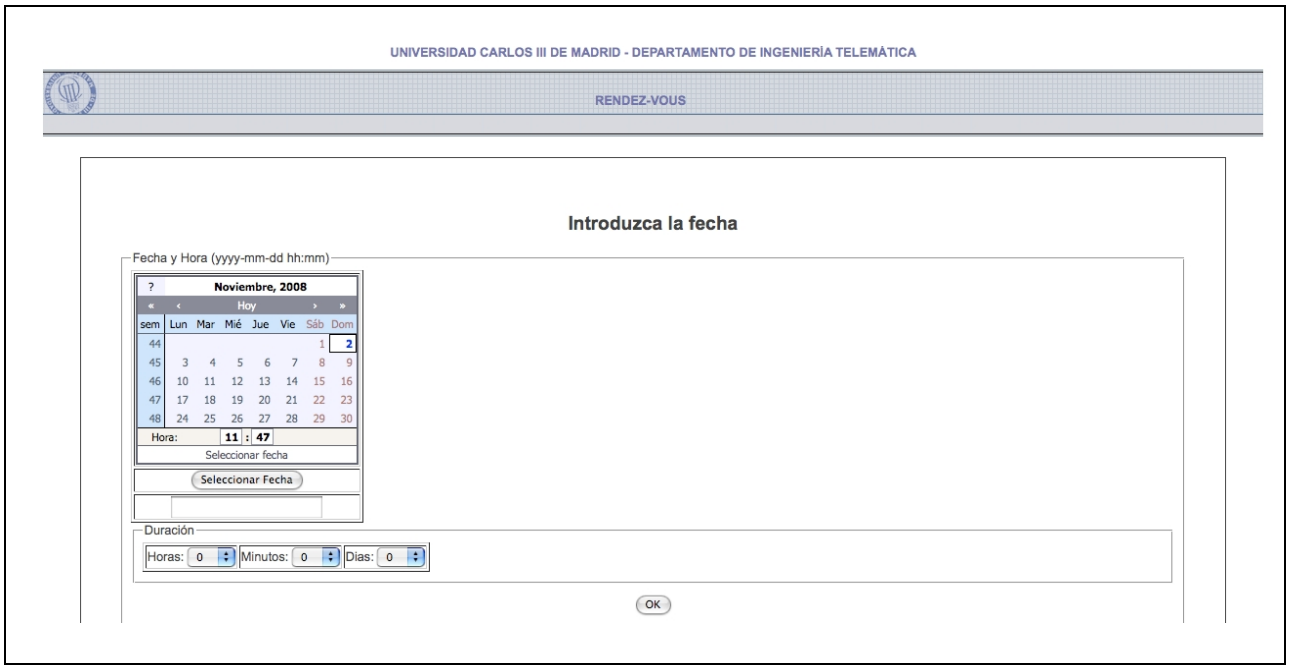

Figura B.4: Calendario para definir una fecha confirmada de una reunión

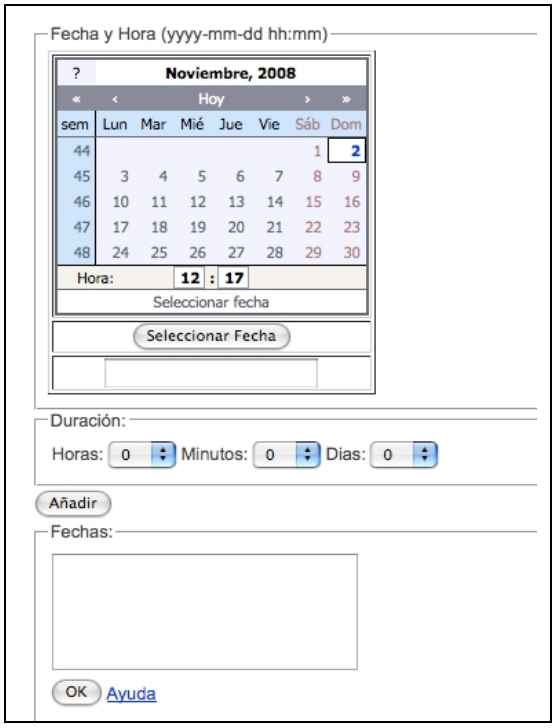

Figura B.5: Calendario para definir las posibles fechas de una reunión no confirmada

Para que no haya ninguna duda de cómo hay que introducir las fechas, la aplicación cuenta con ayudas donde se explica como son los formatos que estas fechas deben cumplir para ser correctas (yyyy-mm-dd hh:mm).

Figura B.6: Modelo de ventana de ayuda de la aplicación

Una vez definidas las fechas, hay que introducir los detalles de la reunión.

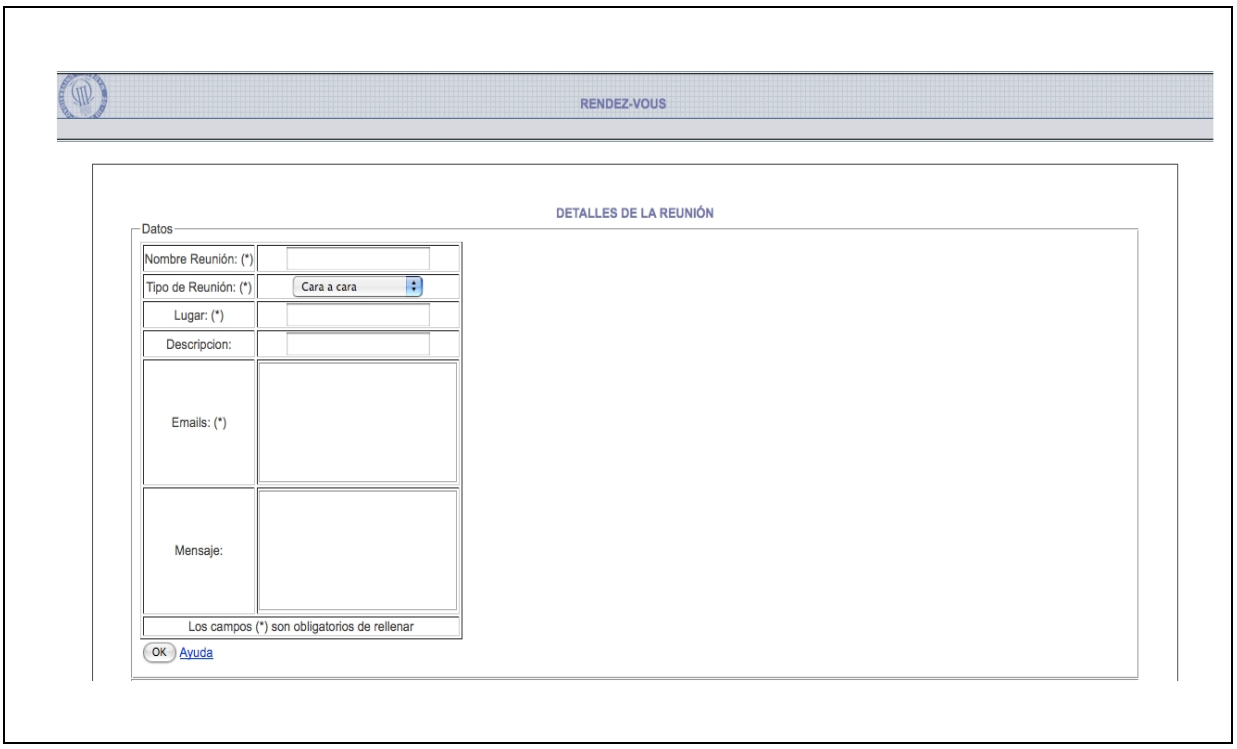

Figura B.7: Cuadro para definir los detalles de la reunión

Nota importante: Las direcciones email de los invitados deben de ir separados por comas (,).

Finalmente se revisan todos los datos y si todo esta bien, se envían las invitaciones vía correo electrónico a cada uno de los invitados.

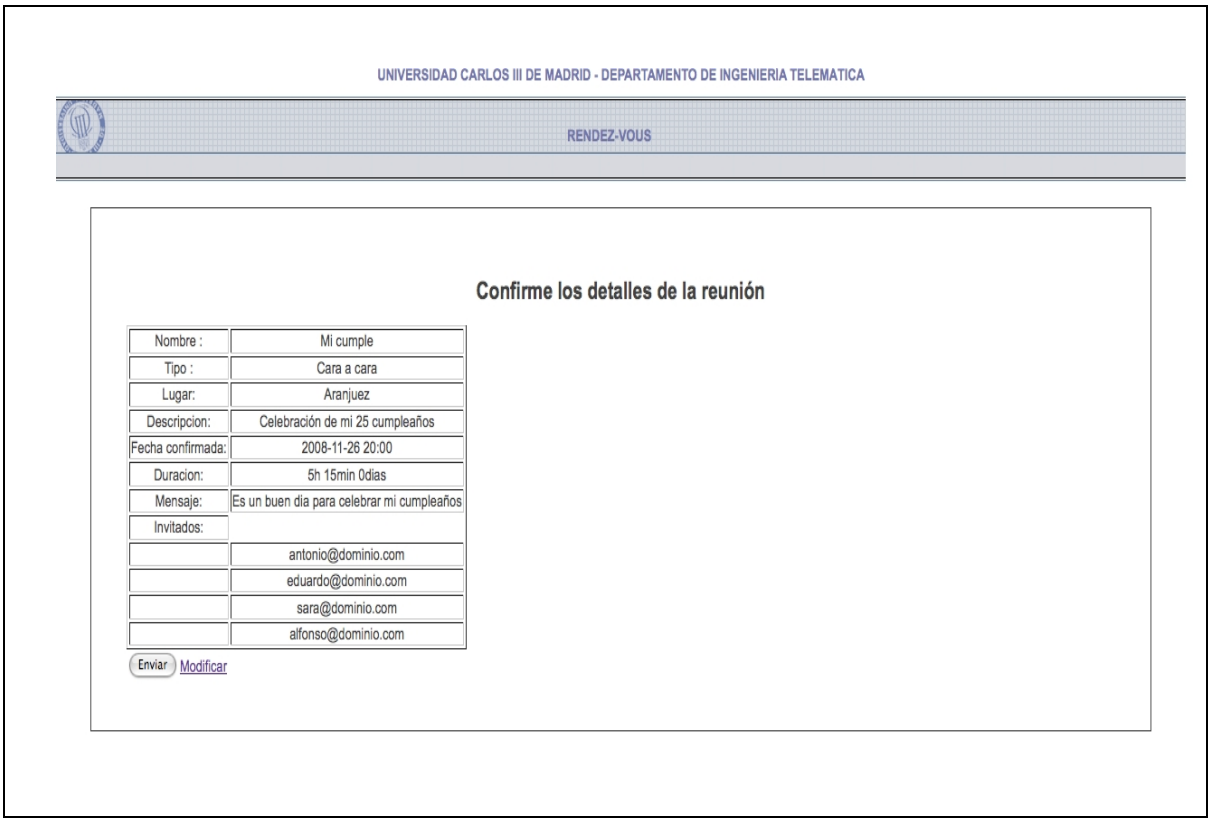

Figura B.8: Cuadro resumen de los detalles definidos para una reunión

B.2 Revisión de las reuniones

Rendez-Vous dispone una opción para que los usuarios puedan revisar las reuniones que han organizado así como a las que ha sido invitado.

Una vez seleccionada una reunión, el usuario cuenta con una serie de opciones que, aparte de detallar los datos de la reunión, muestran el estado de la respuesta de cada uno de los invitados.

A parte de esto, si el usuario es el organizador de la reunión, éste dispondrá de opciones para poder añadir o quitar invitados, fechas así como la posibilidad de poder contestar por alguno de los invitados.

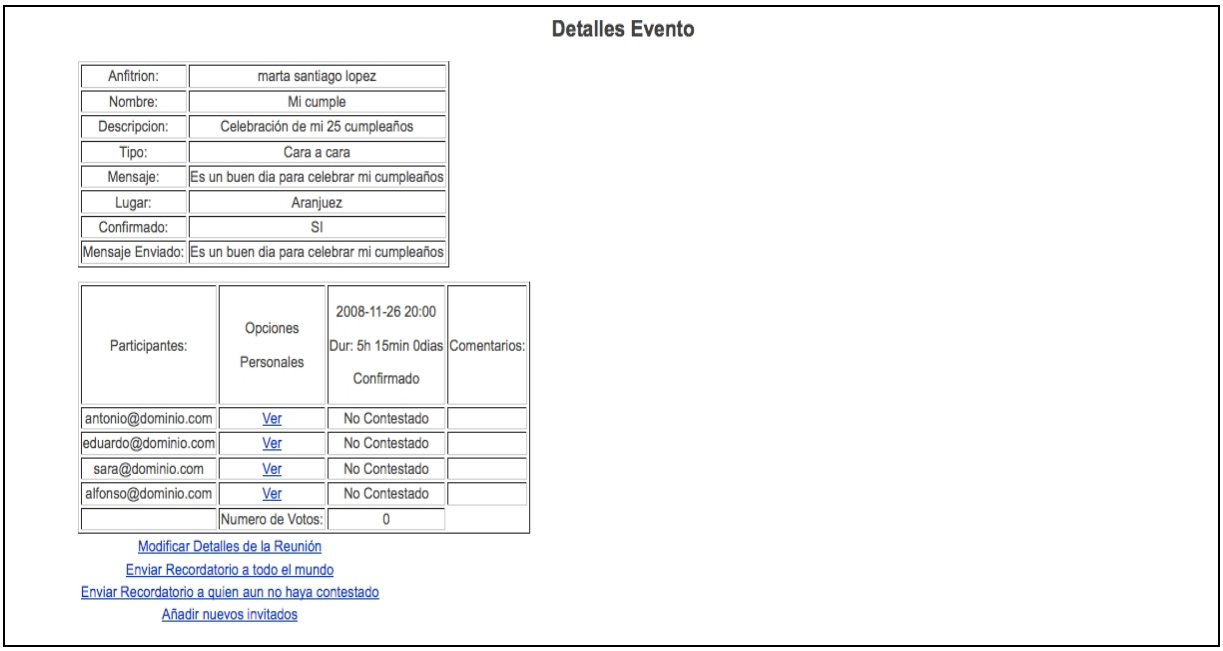

Figura B.9: Cuadro resumen de los detalles y respuestas de una reunión

Por otra parte***, Rendez-Vous*** ofrece en esta misma opción la posibilidad de exportar los detalles de tanto las reuniones creadas como a las que ha sido invitado el usuario. Debido al estándar de ICal, los datos de estas reuniones sólo se incluirán en el fichero **rendezvous.ics** si están ya confirmadas, si no ha pasado la fecha de la celebración de la reunión y en el caso de que se haya sido invitado, el mismo usuario haya confirmado que existirá a dicho evento.

B.3 Votación de las fechas para la celebración de una reunión

Los invitados pueden fijar sus preferencias a través de la URL que reciben en el correo electrónico junto con la invitación. Esta URL les deriva a una página preparada para recoger sus disponibilidades en cada una de las fechas así como un comentario general al evento o a las fechas.

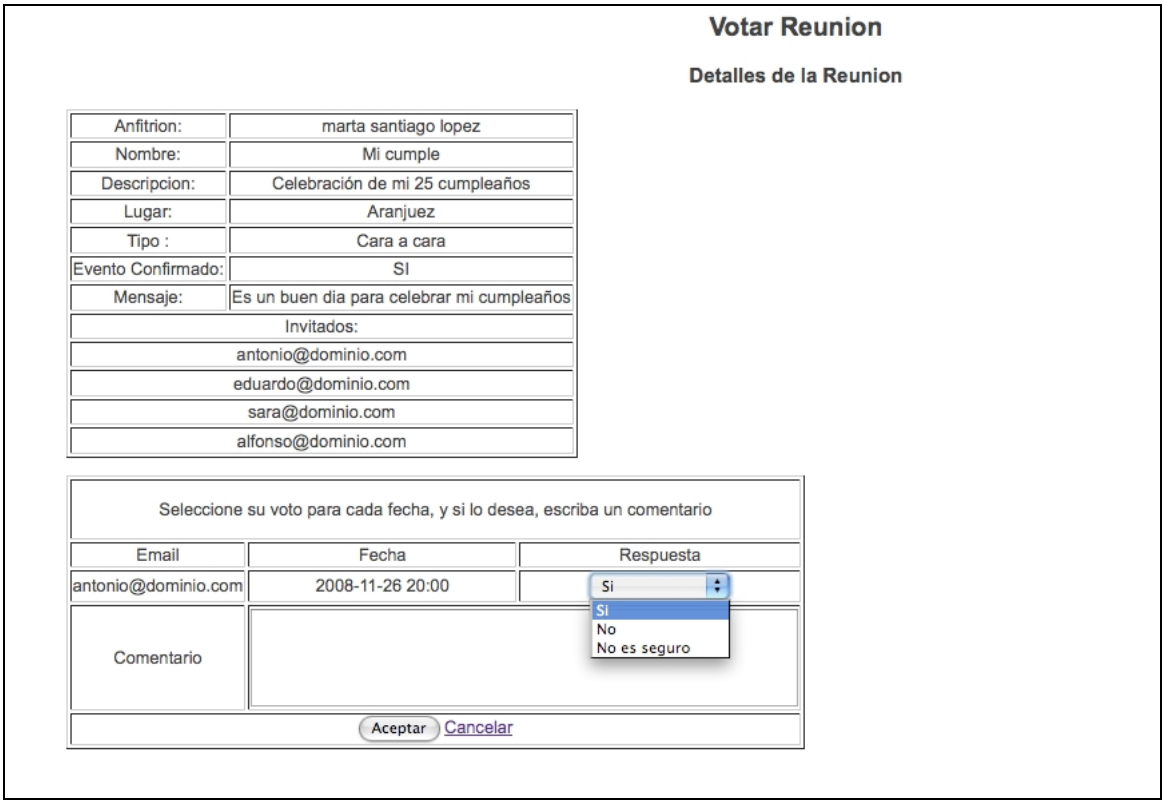

Figura B.10: Página para confirmar las disponibilidades de cada invitado

B.4 Cambio de la información personal

Si en cualquier momento un usuario quiere cambiar su información personal, así como su clave de acceso, Rendez-Vous pone una opción a su disposición para llevar esto a cabo.

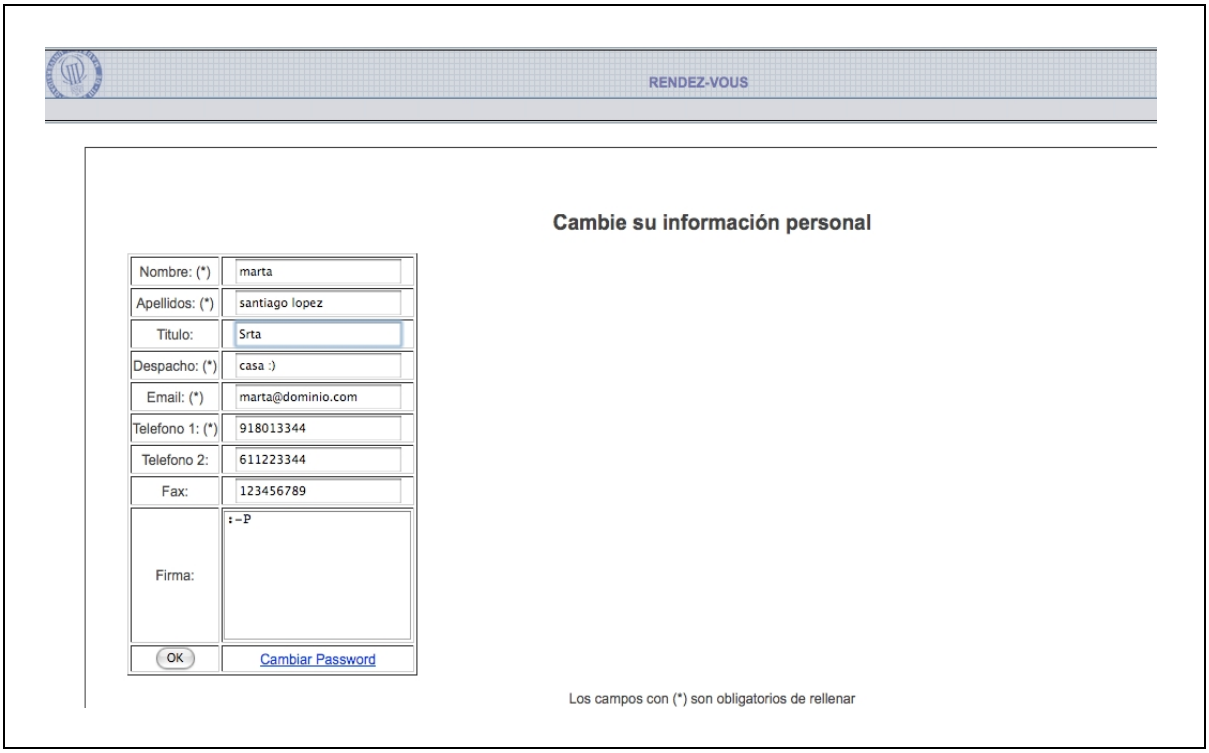

Figura B.11: Ventana para el cambio de la información personal

B.5 Administración de la aplicación

A parte de las opciones anteriormente descritas, que son comunes para todos los usuarios, Rendez-Vous cuenta con ciertas opciones que se usan para la administración de la aplicación. A través de ellas, un administrador puede dar altas y bajas de usuarios en la aplicación, así como restaurar la clave y reenviarla a un usuario si así lo requiere.

Bibliografía y enlaces de consulta

[1] *MeetingWizard:* http://meetingwizard.com/

[2] *Doodle:* http://www.doodle.ch/main.html

[3] *Wikipedia:* http://www.wikipedia.org/

[4] *Java y J2EE:* http://java.sun.com/

[5] *Java:* http://www.programacion.com/

[6] *Servlets:* Aprenda Servltes como si estuviera en Segundo
Javier García de Jalón · José Ignacio Rodríguez · Aitor Imaz, **Universidad de Navarra, Abril 1999**

[7] *J2EE:* Beginning J2EE 1.4: from novice to professional
Weaver, James L.

[8] MySQL: http://mysql.com/

[9] *PHP and MySQL Web development*
Welling, Luke

[10] Apache-Tomcat: http://tomcat.apache.org/

[11] *Encriptación y HASH:*
http://www.devbistro.com/articles/Java/Password-Encryption

[12] *Encriptación y HASH:* http://www.source-code.biz/snippets/java/2.htm

[13] *JavaMail:* http://www.programacion.com/java/tutorial/javamail/

[14] *JavaMail:*
http://www.devjoker.com/asp/impresion_contenido.aspx?co_contenido=1

[15] *XHTML y CSS:* http://www.csszengarden.com

[16] *XHTML y CSS:* http://validator.w3.org/

[17] XHTML and CSS essentials for library Web design
Sauers, Michael P.

[18] *Diseño de páginas Web con XHTML, JavaScript y CSS*
Orós Cabello, Juan Carlos

[19] *Calendario en JavaScript*: http://www.dynarch.com/projects/calendar/

[20] *ICalendar:* http://es.wikipedia.org/wiki/ICal

[21] *Microsoft Web Application Stress*: http://www.microsoft.com/downloads

[22] *Google Calendar:* http://www.google.com/calendar/